5-2012

Development of Oxidizer Flow Control for use in Hybrid Rocket Motors of the Scientific Sounding Rocket Scale

Luke Saindon
lcsaindon714@gmail.com

DEVELOPMENT OF OXIDIZER FLOW CONTROL FOR USE IN HYBRID ROCKET MOTORS OF THE SCIENTIFIC SOUNDING ROCKET SCALE

by

Luke C. Saindon

A Thesis Submitted in Partial Fulfillment
of the Requirements for a Degree with Honors
(Mechanical Engineering)

The Honors College

University of Maine

May 2012

Advisory Committee

Senthil S. Vel, Professor of Mechanical Engineering
Michael Peterson, Professor of Mechanical Engineering
Michael T. Boyle, Associate Professor of Mechanical Engineering
Mark Haggerty, Rezendes Preceptor for Civic Engagement
David Morrison, Assistant Professor of Mechanical Engineering Technology

ABSTRACT

To successfully build a rocket engine with variable thrust you must devise a reliable and robust oxidizer flow control system. The goal of this thesis is to contribute to the goal of building a variable thrust (throttled) hybrid rocket engine, which could eventually be used to power scientific sounding rockets. A variable thrust hybrid engine would increase reusability, flexibility, and capability of almost any small rocket.

Specifically, this thesis work regards the development of the closed loop oxidizer flow control system. To do this, a small test rig was built in the lab that consists of all the components in an actual rocket engine except for the combustion chamber. Using this apparatus the behavior of water in the system was analyzed, including the characterization of the oxidizer flow control valve and the response of the system to various controller parameters. By having a process perfected for characterizing a system with water it makes the process much easier when done with the more exotic oxidizer materials such as nitrous oxide and carbon dioxide.

ACKNOWLEDGEMENTS

Very warm and special thanks to my advisor, Dr. Senthil Vel, who stuck with me through several major adaptions of my topic and goals. There would be no thesis to read without his help, by providing much insight on control systems as well as lending lab space and equipment.

My committee was invaluable as well. Dr. Morrison was excellent help regarding electrical hardware, and spent many hours looking technical spec sheets for me as well as helping with the data acquisition. Dr. Boyle was an excellent source for all thermodynamic questions and provided much calming insight. Dr. Haggerty was excellent outside perspective and kept me on track by knowing the right questions to ask, and was a huge help with the honors reading list. Lastly, Dr. Peterson is owed huge thanks for allowing me to conduct the experiment in Crosby lab, and integrate it closely with my senior design project which he was in charge of.

The development of the thesis topic was inspired by the subject of my senior design project, the building and flying of a sounding rocket. The team I worked with: Alex Morrow, Ryan Means, Robert Miller III, Josh Mueller, and Gerard Desjardins were all incredibly patient to allow me to work on an honor's thesis is parallel with our senior project. Many thanks go to them as well.

Lastly, the professionals from the Mavericks Civilian Space Foundation, Marshall Space Flight Center, Ames Research Center, Space Propulsion Group, Maine Space Grant Consortium, and Applied Thermal Sciences are all owed a warm thank you for helping me with the design process and educating me regarding hybrid motors.

TABLE OF CONTENTS

Chapter 1: introduction and Motivation 1

1.1 Final Goals of Thesis 2

1.2 Layout of this Thesis 4

1.3 Hybrid Motor Basics and Background 5

1.4 Hybrids and throttling, Flow Control Technology and its challenges 8

Chapter 2: Introduction to control systems 11

2.1 PID controller basics 12

2.2 Implementing an Oxidizer Flow Controller 14

Chapter 3: Test Setup: Description of Apparatus 16

3.1 Tanks and Piping 16

3.1.1 Control Valve Assembly 23

3.1.2 Oxidizer Tank 25

3.2 Instrumentation and Data Acquisition 30

3.3 Description of LabVIEW Program 33

3.4 Description of Arduino Program 37

3.4.1 Manual Arduino Program 38

3.4.2 Valve Characterization Program 40

3.4.5 Closed Loop Flow Control Program 41

Chapter 4: Modeling the system 43

4.1 Assumptions about the system 43

4.2 Governing equations 45

4.3 Modeling the valve response (effects of delay) 48

4.4 MATLAB simulation description and explanation 48

Chapter 5: Testing Procedure 54

5.1 Characterizing the Valve 54

5.1.1 System Preparation from a Cold Start 54

5.1.2 Running a Test 56

5.1.3 Resetting for another Test 56

5.1.4 System Shutdown 57

5.2 Testing the Control System 57

5.2.1 System Preparation from a Cold Start 58

5.2.2 Running a Test 58

5.2.3 Resetting for another Test 58

5.2.4 System Shutdown 58

Chapter 6: Post Processing 59

6.1 VF Plot Post Processing 59

6.2 Control System Analysis Post Processing 67

Chapter 7: Results 69

7.1 Valve Characterization 69

7.2 Control System Results.....74

Chapter 8: Discussion of results 82

Chapter 9: Conclusion.....85

Appendix A: Simulink Model Screenshots……87

Appendix B: VF Plot Post Processing MATLAB Code……90

Appendix C: Controller Analysis Post Processing MATLAB Code…….94

Appendix D: Manual Arduino Code……..99

Appendix E: Specific Angle Arduino Code……105

Appendix F: Controller Arduino Code……109

Appendix G: Technical Data Sheets……116

Appendix H: Design Summary of a Possible Hybrid Motor……...134

Appendix I: Technical Drawings…….168

Biography of the Author……..201

TABLE OF FIGURES

Figure 1: P&ID Diagram of Experimental Setup 18

Figure 2: Photograph of Apparatus 20

Figure 3: CAD Rendering of Apparatus 20

Figure 4: Photograph of Nitrogen Fill Station 21

Figure 5: Photograph of Upper Tank Bulkhead 22

Figure 6: CAD Rendering of Valve Assembly 24

Figure 7: Photograph of Valve Assembly 25

Figure 8: CAD Rendering of Oxidizer Tank End 28

Figure 9: CAD Rendering of Upper Oxidizer Tank Bulkhead 29

Figure 10: CAD Rendering of Lower Oxidizer Tank Bulkhead 29

Figure 11: System Instrumentation and Wiring 31

Figure 12: Motor controller 32

Figure 13: Step motor power supply 32

Figure 14: Arduino micro controller 32

Figure 15: 5VDC power supply 32

Figure 16: NI chassis and modules 32

Figure 17: DAQ Block Diagram 34

Figure 18: DAQ Front Panel 35

Figure 19: Screenshot of DAQ Assistant Configuration 36

Figure 20: Screenshot of Arduino Compiler 38

Figure 21: Bernoulli's Streamline 46

Figure 22: Overall Block Diagram 49

Figure 23: Controller Block Diagram 49
Figure 24: Simulation Plant Subsystem 51
Figure 25: Tank Simulation Subsystem 52
Figure 26: Mass Curve Noise 62
Figure 27: Mass Flow Curve Noise 63
Figure 28: Temperature Noise 64
Figure 29: Pressure Noise 65
Figure 30: Step Command Noise 66
Figure 31: Direction Command Noise 66
Figure 32: Enable Command Noise 67
Figure 33: VF Plotted Against Valve Position and Pressure Drop 70
Figure 34: XY Plane of VF Surface 70
Figure 35: XZ Plane of VF Surface 71
Figure 36: YZ Plane of VF Surface 71
Figure 37: VF vs. Valve Position dP=10psi 72
Figure 38: VF vs. Valve Position dP=20psi 72
Figure 39: VF vs. Valve Position dP=40psi 72
Figure 40: VF vs. Valve Position dP=85psi 72
Figure 41: VF vs. Pressure Drop POS=35deg 73
Figure 42: VF vs. Pressure Drop POS=55deg 73
Figure 43: VF vs. Pressure Drop POS=75deg 73
Figure 44: VF vs. Pressure Drop POS=90deg 73
Figure 45: System Mass Flow Response for KI=20, KP=20 75

Figure 46: System Mass Flow Response for KI=5, KP=10 75

Figure 47: System Mass Flow Response for KI=2, KP=10 76

Figure 48: System Mass Flow Response for KI=5, KP=2 76

Figure 49: System Pressure Response for KI=5, KP=10 77

Figure 50: System Pressure Response for KI=2, KP=10 77

Figure 51: Variety of System Mass Flow Responses 79

Figure 52: Variety of System Pressure Responses 80

LIST OF EQUATIONS

Equation 1: PID Controller.....12

Equation 2: Noise Amplification Upon Differentiation.....13

Equation 3: Determining Step Interval from Valve Angular Velocity.....42

Equation 4: Mass flow based on pressure drop and valve position.....46

LIST OF TABLES

Table 1: System P&ID Diagram.....18

Table 2: Control Valve Design.....24

Table 3: Instrumentation Details.....31

Table 4: DAQ Inputs and Outputs.....37

VARIABLE DEFINITIONS

KI	Integral Proportionality Constant
KP	Proportionality Constant
θ	Valve Angular Position
$\dot{\theta}$	Valve Angular Velocity
$\dot{\phi}$	Motor Angular Velocity
ϕ	Motor Angular Position
N	Number of motor steps
KI	Controller Integral Proportionality Constant
KP	Controller Proportional Constant
KD	Controller Differential Proportionality Constant
L	Valve Factor Multiplier
VF	Valve Factor
D_{Loss}	System Loss Coefficient
ρ	Density
P	Pressure
ΔP	Pressure differential (pressure drop across the system)

CHAPTER 1: INTRODUCTION AND MOTIVATION

The ability to throttle a rocket motor increases its flexibility such that it can be used in a variety of applications, and performs more efficiently. Throttling liquid propellant motors has been perfected and executed already; the SSME's (Space Shuttle Main Engines) throttle from 64-109%. Solid propellant motors have no throttling or shutdown capabilities, but are simple and robust. Hybrid rocket motors use a fuel and oxidizer combination of two different phases, most commonly a solid fuel and liquid oxidizer. They have much of the simplicity of a solid motor with more safety and with the potential to be as controllable as a liquid motor. Building a simple thrust control system for a sounding rocket sized hybrid motor would be a boon for scientific rocketry since many payloads require a gentler launch than can be achieved with a conventional solid motor, which is the most common vehicle at this time. Liquid propellant motors provide throttling, but they are undesirable at the scientific sounding rocket scale for other reasons such as: complexity, cost, and higher mass fractions.

Sounding rockets are smaller launch vehicles that carry scientific payloads to sub-orbital altitudes, usually in the 100,000 ft range and higher. The ultimate launch method for some of these payloads is one that, as mentioned above, is gentler in the ascent. Solid rocket engines tend to start with a hard kick and generate extremely high g-loading during flight, as high as 20-30 g's. If a throttled hybrid motor was developed then the user of the rocket could program the motor's control system to produce a desired thrust curve which would minimize the payload loading produced by hard acceleration. Motors can produce equal amounts of impulse but in very different ways; either high thrust for a short duration or low thrust for a long duration. The latter produces the smoother ride

into the upper atmosphere, while also usually achieving a higher efficiency (less fuel for an equal altitude). In simple terms, it takes a lot of energy to push through the thick lower atmosphere at a high speed. Much like it takes less energy to walk calmly through a pool instead of attempting to run. By precisely tailoring the thrust curve using a control system on a hybrid rocket engine would allow the flight profile to be highly optimized, allowing higher altitudes for the same quantity of fuel, while avoiding the expense and complexity of a full liquid system.

1.1 Goals of the Thesis

The objective of this honors thesis is to help develop such a throttled hybrid rocket motor for use on scientific sounding rockets. While building a complete rocket engine was the original plan, it soon became apparent that we did not have the facilities necessary to safely ground test such an engine in Maine. Instead, a mockup of the complete oxidizer flow path, from the oxidizer tank through the control valve and all the way into the injector was developed in the lab and only the response of the oxidizer system was investigated.

This honors thesis investigates the process of feeding an oxidizer from a holding tank, through a control valve, and into the combustion chamber injector. A computer model of this oxidizer feed system will be built in Simulink and then the actual hardware will be tested in the lab to fine tune the many parameters of the computer model such that reality matches prediction. Simulink is a block diagram based numerical solver, and is a component of MATLAB. Water will be used in the place of the oxidizer, but if the process of system characterization is repeated any fluid could be modeled with equal

accuracy. Since a hybrid rocket engine is throttled by controlling the flow of oxidizer through this feed system, a controller will be implemented in the lab setup. In summary, the goals of this thesis will include:

- Create an accurate computer model of the oxidizer feed system. The largest unknown in the system is the flow control valve itself. Therefore a large component of this thesis work is to fully characterize this valve.
- Develop a controller used to monitor and adjust the oxidizer mass flow. Once the flow control valve is characterized the system can be simulated in the computer, parameters for a control system can be determined, and then the flow control system can be tested in the lab to see how closely the computer simulation matches reality.

Additional goals include documenting the most efficient and effective methods of performing the above goals such that new oxidizer fluids can be quickly characterized and a control system tuned to effectively control the flow of the new fluid; and draw parallels between the lab bench test and applications on actual hybrid rocket engines such that this thesis investigation will be useful in the development of flight engines.

Understanding the way water behaves in a system identical to that which would feed an actual combustion chamber with an oxidizer is an excellent base of knowledge that would be necessary to build this ideal throttled hybrid engine. All of this will be done at the same scale and in way that would be possible in an actual sounding rocket. By working within the size limits of the sounding rocket airframe it is possible to build a motor with a conservative thrust estimate of 1200 lbf; delivering a total of 120,000 Ns of

impulse. For perspective that would put the engine in the Q scale range of rocket motor, if one is familiar with the traditional nomenclature of rocket motor sizing. Most little model Estes rockets are in the A-D range, where each subsequent letter has double the impulse of the one before it. A well-designed control system must allow the hybrid to be throttled within a 3:1 range (the minimum thrust would be a third of the max thrust). Total burn time would be a maximum of 30 seconds, governed mostly by the limited oxidizer capacity.

Interestingly, in parallel to this project a Mechanical Engineering senior design project is building a sounding rocket that requires an engine of similar size. Therefore all of the experimental apparatus has been designed with a hybrid motor which could power this sounding rocket in mind. Later in this written thesis Appendix H looks at the design and simulation of a complete motor, not just the oxidizer feed system.

1.2 Overview of the Thesis

Now that some foundation as to what motivated this thesis has been presented it is advantageous to understand the general layout of this written document. First, a more detailed view of how hybrid motors work will be presented; as well as how oxidizer flow control could be implemented on an actual motor. Throughout these discussions many examples of current hybrid projects as well as a state of the art will be given.

Focusing more on the contents of this thesis regarding the flow control itself an introduction and explanation of control systems will be given, as well as how a control system will be implemented in this project. A review of the PID controller specifically and how it is used in this thesis will be covered.

In order to meet the goals set, an experimental apparatus and supporting software must be developed. A detailed description of the test apparatus will be next, explaining the hardware and software as well as the data acquisition system used. The experimental results will be used to fine tune the simulation model, this data will be displayed and described in the results section. No model is useful without verification, no matter how fancy the simulation process there needs to be some data somewhere to back it up.

The difficult part of the thesis involves the accurate modeling of the system, particularly of the control valve. What it means to characterize the flow control valve will be presented; as well as the methods used to simulate and model the system as a whole within MATLAB. The assumptions made in the simulation process are also very important to understand, since any real system is much too complicated to model in perfect exactitude. Therefore the assumptions will be made clear.

1.3 Hybrid Motor Basics and Background

For perspective it is important to have some basic understanding of the operation of hybrid motors, this will also help demonstrate how the fluctuation of oxidizer flow could result in the fluctuation of thrust. Understanding some basics about these motors helps put the thesis work in perspective. As explained earlier, a hybrid motor uses a liquid oxidizer and a solid fuel to create the necessary chemical components for combustion within the combustion chamber. Theoretically it is possible to have a solid oxidizer such as ammonium perchlorate and a liquid fuel instead, but very little work has been done with this. Hybrids, in general, are much less understood than either liquid or solid propellant motors. Their main appeal is to be safer than solids, cheaper and simpler

than liquids; however they maintain much of the throttling ability and adaptability of liquids. There are disadvantages as well, and will be covered shortly.

The oxidizer is fed from a holding tank through an injector which atomizes the oxidizer inside the combustion chamber. As the oxidizer moves across the surface area of the solid fuel grain a thin layer of fuel vaporizes and allows combustion to be sustained. Some ignition source of enough energy to begin vaporizing the solid fuel must be used before oxidizer is introduced and the engine throttled up. One of the largest disadvantages of hybrids is the limited source of fuel. The solids can only vaporize at the exposed surface and can only do so at a finite rate for a given amount of oxidizer. The speed at which the solid fuel can vaporize for a given mass flow of oxidizer is called the fuel regression rate, and it is very important to understand the characteristics of a fuel and how it burns with a specific oxidizer before a motor can be confidently and safely designed. Engineers are constantly attempting to find new ways of increasing the regression rate of fuels such that the surface area can be reduced and total fuel volume increased to create more compact high thrust motors. One option is to use a wax based fuel, which instead of vaporizing from a solid state at a fairly limited rate first melts into a liquid boundary layer. This liquid layer can actually be caught up in the gas streams headed to the nozzle and become airborne while burning; effectively increasing the surface area of fuel exposed to oxidizer. Much like spray can be picked up from the tips of white cap waves on a stormy day. Paraffin fuels are now being investigated as a primary fuel source for many hybrid engines. If implemented correctly, paraffin could nearly negate the regression rate problems of other fuels such as HTPB, and provide

ample fuel for a given flow of oxidizer. There are still a lot of teething problems with paraffin hybrids, largely burn stability.

The use of hybrid rocket engines is fairly limited at this point in time, but they are becoming more popular and used in a wider range of vehicles. Possibly the most famous application was in SpaceShipOne, built by Scaled Composites. This vehicle is designed for sub-orbital flights up to 100 km, and is powered by a Helium supercharged Nitrous Oxide and HTPB hybrid motor. They successfully competed for the Ansari X-prize in 2004 for launching a manned aircraft to an altitude greater than 328,000 ft and doing so twice in less than 14 days. Scaled Composites is currently working on a SpaceShipTwo and has partnered with a Richard Branson company called Virgin Galactic to provide tourist service to suborbital space. SpaceShipTwo will use a much larger version of the hybrid motor in its little brother.

Another demonstration of hybrid propulsion will be with Sierra Nevada's Dream Chaser vehicle. The Dream Chaser will use two hybrid motors to boost into LEO (low earth orbit) to provide cargo and shuttling service to the ISS. The hybrid motors are those used in SpaceShipOne. SpaceDev is responsible for the motor's design and is actually a component of the Sierra Nevada Corporation. These particular hybrid motors are not throttled, and are using the better understood, but lower performance, HTPB as fuel. If a paraffin version of these motors could be developed and a throttling system implemented the hybrid motor would become a much more appealing option for many other vehicles.

A variety of sounding rockets and other conventional rocket vehicles have tried using hybrid propulsion, all with varied success. Some failures could have been easily

avoided, such as Amroc's stuck oxidizer valve which caused the oxidizer to only slowly flow into the combustion chamber. This allowed enough flame to incinerate the rocket without moving it an inch. A team from New Zealand just launched a hybrid sounding rocket, but failed to recover the second stage, no fault of the hybrid, however.

At the moment, one of the most promising efforts is being led by Stanford University with support from NASA Ames research center. The so called Peregrine project will be a 100 km sounding rocket powered by a 10 in diameter paraffin and liquid oxygen hybrid. The project is also getting support from Space Propulsion Group, the industry leader in paraffin hybrids. As of yet the motor has no closed loop thrust control system.

1.4 Hybrids and throttling, Flow Control Technology and its challenges

Throttling rocket engines is not a trivial affair, no matter the engine. Solid propellant motors cannot be throttled, once lit they burn until all the fuel is consumed. Some control over the thrust curve can be achieved by being clever with grain design or fuel composition variations radially within the grain; such that the exposed surface area or volatility of the fuel changes in a certain way over time thus changing the thrust curve. There certainly is no shutting a solid motor down once ignited.

Throttling liquid propellant motors is the most common and best understood at the present. In a liquid engine you have direct control over the flow of oxidizer and the fuel, by using valves in the feed lines. To reduce thrust you must reduce the chamber pressure. Reducing pressure in the combustor is as simple as reducing the amount of fuel and oxidizer fed into the system. Usually to burn at maximum efficiency the motor requires

more oxidizer than fuel (by mass) therefore the flow of oxidizer has to be cut more than the flow of fuel in order to maintain ideal operating conditions. With a liquid propellant engine you can control this oxidizer to fuel ratio (O/F) directly, allowing the engine to operate as efficiently as possible at any given thrust level.

Hybrid motors, since they use a solid fuel and a liquid oxidizer, are a little less precise when it comes to throttling. The operator only has control of the oxidizer flow, and the fuel flow is a function of oxidizer flow. This goes back to the regression rate idea that was discussed earlier. Now that this is a coupled system the change in oxidizer flow also means a change in fuel flow. In a simple sense, cutting the flow of oxidizer cuts the flow of fuel because the regression rate is a function of mass flux in the fuel port. However, usually the motor will only operate at ideal O/F ranges for one specific mass flow of oxidizer. At all other throttled points the motor will not be running at maximum efficiency. This is a tradeoff. The hybrid offers increased simplicity, and lower costs than a liquid propellant motor, but does not have the potential to be quite as efficient at all thrust levels. It is a mid-point between solid motors and liquid motors. Further exploration into the specific design of a hybrid motor is presented in Appendix H, and looks at the O/F tradeoffs in more detail.

Hybrids, as well as most other rocket motors are currently thrust controlled by varying the flow of fuel and oxidizer manually, and does not use a closed loop control system to verify that the engine is providing the correct thrust. In other words, a desired thrust curve is achieved by spending a lot of time characterizing a specific motor such that the operator knows that if you specify a curve of oxidizer mass flow over time then you will get a desired thrust curve over time. This is fine as long as you don't change the

desired curve, and that the motor is operating exactly like it was designed and characterized to run.

If a hybrid motor could be fitted with a controller that measures thrust, computes how different the thrust is from a reference thrust curve, and then adjusts the flow of oxidizer accordingly, then you could have a much more robust and versatile system. It is a method of control that is used in a variety of other engineering applications, but never routinely on launch vehicles to allow the launch team to optimize the thrust curve. The ultimate scenario is that this hybrid is developed such that a team interested in using it as a booster for a payload could determine the optimized thrust curve which maximizes altitude or some other parameter while minimizing the cost of launch (i.e. by using less fuel). In theory, once the thrust curve has been determined it can be loaded into the booster computer memory and the PID control system will enable the motor to follow the curve by using a closed loop to constantly monitor the actual system performance. If the motor is reliable and stable at a wide range of thrust levels then this should be a fairly simple task. The work of this thesis investigates the performance of a specific tank/valve/driver/injector setup in controlling the mass flow of water. Having this experience will be invaluable when it comes to designing the rest of the motor. The idea of having the capability of loading any curve into the control system at a moment's notice and having the motor follow that curve is an exciting prospect. Other possibilities include real time control of thrust via radio link or some other input. The robust nature of a closed loop control system opens all sorts of exciting possibilities.

CHAPTER 2: INTRODUCTION TO CONTROL SYSTEMS

A well designed control system can be one of the most elegant systems encountered in engineering. They add a huge amount of functionality and usability to many designs. Through the use of a controller, a system can be forced to almost perfectly follow a desired reference signal. However, what makes the use of a control system special is that it has no high level knowledge of the system which it is controlling. It uses very simple logic to create extremely complex results, making it very robust.

The general idea is that there is some reference curve that you want your system to adhere to. An example would be the angular velocity of an electric motor over time. Perhaps you want the angular velocity to follow a specified curve, regardless of the load on the motor. If you knew the load to be applied you could feed the motor a pre-determined voltage vs. time signal that would keep the angular velocity at the desired level. This, however, is a very fragile way of executing the task; what if some parameters in the system change? It would be inconvenient to re-characterize the motor input anytime this happened. Instead, use a controller that constantly compares the actual motor rpm with the reference curve; resulting in an error value. The controller uses this error value to either increase or decrease the input to the system (voltage to the motor in this case). If the motor slows down due to a load then the controller reads an rpm lower than reference and gets a larger error value. This large error results in the controller increasing the voltage to the motor. If the error was opposite in sign then the controller would decrease the voltage to the motor.

2.1 PID controller basics

The next question is how the controller arrives at a system input based on an error signal. It is important that the controller does not actually know anything directly about the system when it goes about determining this input value, since that would increase complexity of the controller, increasing error and decreasing the robustness. There are three main means of computing the system input: direct proportionality to error; derivative proportionality to error; and integral proportionality to error. The term "PID controller" may sound familiar; where the P stands for proportional, the D for derivative, and I for integral. One does not need to use all three methods, often just a proportional controller is necessary, or a PI, or even a PD controller. Each of the methods has its benefits and drawbacks. Equation 1 shows how the controller output signal is generated from the error term.

$$input(t) = (KP * e(t)) + \left(KI * \int e(t)dt\right) + \left(KD * \frac{d}{dt}e(t)\right)$$

Equation 1: PID Controller

If only a proportional multiplier is used then the system will tend to always have an offset from the reference. This is because unless the proportionality constant is very large then as the error goes to zero, the input the controller delivers to the system decreases to zero as well. Some equilibrium point is found, but there will always be some error between the reference and response. Proportional controllers are very simple and stable, however.

An integral controller is used to get rid of the offset that is developed with only a proportional controller. This makes a PI controller very common. The integral method

integrates the error over time. Therefore the offset generated by the proportional controller will slowly build up over time and the integral component will see this as a need for a larger system input and thus draws the actual system output closer to the reference signal. The problem with integral controllers is their tendency to overshoot the reference and then oscillate about it for a significant amount of time.

The overshoot problem is where the derivative controller comes in. The derivate component senses how quickly the error value is changing. If the error is increasing quickly then the derivative portion of the controller will increase the input more drastically to counter the increasing error. On the flip side, when the error is actually quite close to the reference signal and not changing all that fast the derivative controller calls for gentler and less dramatic system input. They are very useful for systems with a lot of momentum that like to overshoot and maintain oscillations for long durations. The trouble with derivative controllers is that they are very sensitive to noisy system output, and can make the controller output (system input) an amplified version of the system output that quickly spirals out of control. Careful filtering is necessary to use a derivative controller effectively. If you look at Equation 2, where y(t) is a noise component of an error signal being fed to the controller and ω is large since the noise is usually of high frequency, you can see that differentiating it creates large amplitudes, $\omega\varepsilon$, that can cause erratic and undesirable controller output to the system.

$$y(t) = \varepsilon \sin(\omega t)$$
$$\frac{d}{dt} y(t) = \omega\varepsilon \cos(\omega t)$$

Equation 2: Noise Amplification upon Differentiation

2.2 Implementing an Oxidizer Flow Controller

For this thesis a controller will be used to monitor and correct the mass flow of oxidizer. In this case the controlled quantity is the "oxidizer" (water) flow through the system. There will be a reference curve of mass flow vs. time and the controller will attempt to maintain the system so that the actual mass flow matches the reference mass flow rate. In order for the controller to compute an error value it must have a reading of the system output (mass flow). This will be accomplished by hanging the oxidizer tank from the ceiling with a load cell. The weight of the tank can be differentiated over time to get the mass flow out of the system. The controller will then take this error value and run it through a PID controller to get an input for the system. The controller output or system input will be an angular velocity of the valve (ie CCW at 95 rpm). When the motor responds to this and starts changing the valve position at this particular rate then the oxidizer flow will change and the controller will reassess the situation. Hence the term closed loop control.

Eventually, when this oxidizer flow control is attempted on a throttled hybrid motor the only difference will be how the mass flow of oxidizer is measured. Instead of having a direct measurement from a load cell, the controller will use a chamber pressure measurement from the combustion chamber. From this pressure measurement a good approximation of motor thrust can be determined, and this motor thrust can be then compared to a thrust vs. time reference curve to generate an error signal which the controller can use. Thrust is primarily a function of oxidizer mass flow (there are some other minor factors), therefore the thrust control and the oxidizer mass flow problems are

very interrelated. If a controller for mass flow is developed then there is a lot of necessary knowledge that can be used for the application on a working motor.

CHAPTER 3: TEST SETUP: DESCRIPTION OF APPARATUS

The oxidizer feed apparatus that is setup for this experiment represents a possible system which could be implemented on a hybrid motor of the size to be used in the Ursa class sounding rocket (Project Ursa is the name given to the University of Maine sounding rocket project). Much of the apparatus is designed and built as it would appear on an actual rocket motor. The tank is constructed to size and shape of a possible flight tank, and is hooked up to a flow control valve operated by a stepper motor. This stepper motor and valve were designed to consume power and fit in space that would be available onboard the vehicle. An injector is mounted aft of the control valve. The only missing element is the combustion chamber itself. Technical drawings of all the components are available in Appendix I, which presents the plans for a hypothetical hybrid motor and therefore includes other components in addition to the oxidizer tank.

3.1 Tanks and Piping

The injector is a multiport showerhead type, with a 1" NPT inlet. At 300 psi of pressure drop across the valve and injector the predicted flow of water through the system is 3.3 kg/s or about 65gal/min. These flow capacities are sufficient for the proposed rocket motor design presented in Appendix H. Since water will be tested at first and is not self-pressurizing at atmospheric conditions, nitrogen gas will be used to create the necessary pressure drop between the tank and the atmosphere. Table 1 displays a P&ID (Piping and Instrumentation Diagram) of the system as it is constructed for this experiment. A computer rendering and photograph of the actual test rig are presented in

Figures 2- 5 along with some of the basic components labeled. The specification sheets for the major components can be found in Appendix G.

The nitrogen fill and the water fill locations as well as a load cell are on top of the oxidizer tank. The nitrogen fill bottle sits strapped to an I-beam nearby and fills the oxidizer tank through a length of nylon pressure hose. The nitrogen fill has a pressure relief valve with a higher flow capacity than the regulator and has a lower relief pressure than any component's pressure rating. At the nitrogen supply bottle a regulator reduces the pressure from the tank to the desired system pressure; two gauges allow the tank pressure and the supply pressure to be read. Asides from the main tank shutoff there are two additional valves at the nitrogen tank, one shuts off the nitrogen supply to the oxidizer tank and the other allows a purge of the system. Purging the system is necessary either when completing a set of tests, or while refilling the oxidizer tank with water since the ullage gasses need somewhere to go while being displaced with water.

At the top oxidizer tank bulkhead there is a valve that shuts off the water supply, this valve is necessary so that when the tank is pressurized by nitrogen the garden hose which supplies the water isn't also pressurized since it isn't a rated component. The load cell is attached to the center of the bulkhead and allows the tank to hang straight. On either end of the load cell ball joints are used so that the cell can only be put in perfect tension. From the ball joints the tank is shackled to a chain which runs over a ceiling truss.

Table 1: System P&ID Diagram

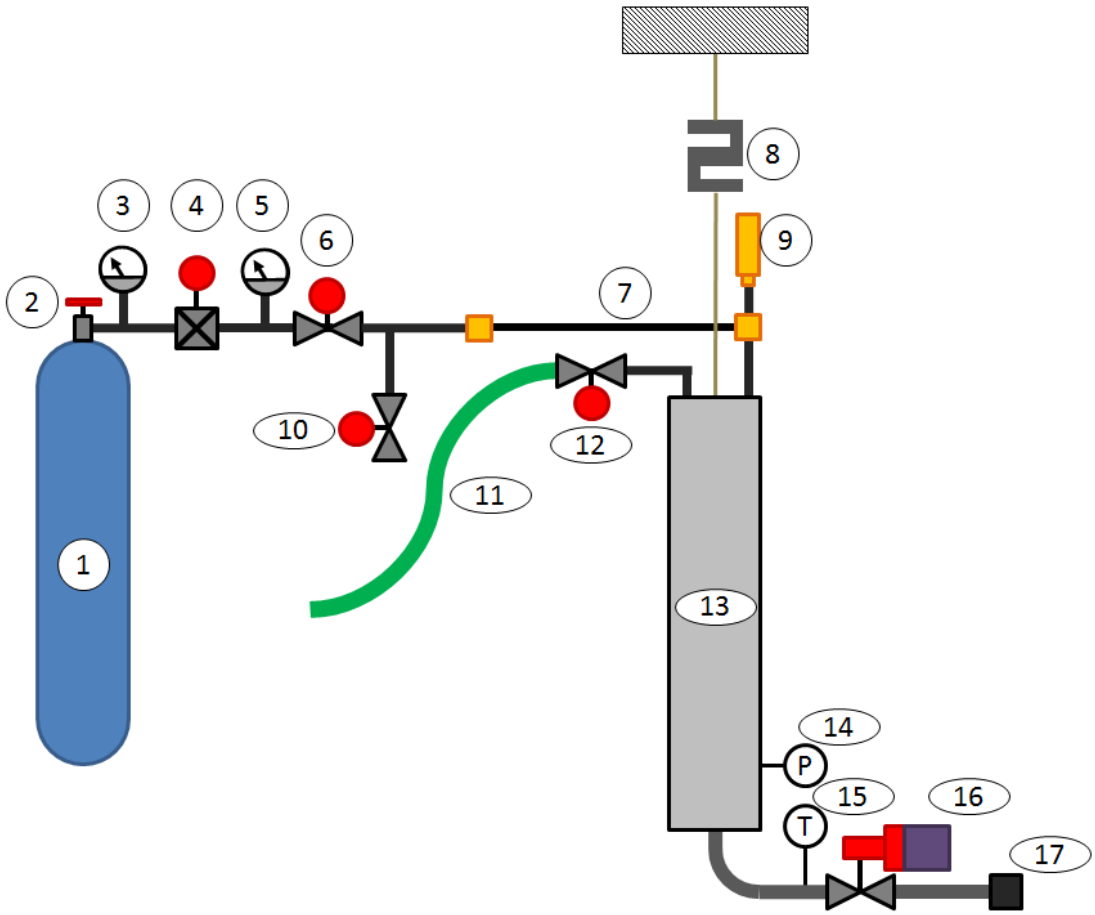


Figure 1: P&ID Diagram of Experimental Setup

1	DOT bottle containing gaseous nitrogen • initially at 2500psi • Type K • 244 Cu ft. of Nitrogen at STP Conditions
2	Main bottle shutoff valve • CGA-580 Connection
3	Nitrogen bottle pressure gauge • 0-4000psi Range
4	Nitrogen supply pressure regulator • 0-500psi Capability
5	Nitrogen supply pressure • 0-500psi Range
6	Pressure shutoff valve • 3000psi Rated • 3/8” FNPT

	• Ball Type
7	Flexible nylon pressure hose • 750psi Rated • 3/8” OD Nylon
8	Load cell • 0-500lb • See Appendix G for details
9	Pressure relief valve • 750psi set pressure
10	System purge valve • Identical to item 6
11	Water fill hose • Garden Hose
12	Water fill valve • Identical to item 6
13	Oxidizer tank
14	Oxidizer tank pressure transducer • 0-1000psi range • See Appendix G for details
15	Oxidizer flow thermocouple • Type T • Ungrounded • Open Tip
16	Step motor driven oxidizer flow control valve (see next section for more details)
17	Injector • 1” FNPT • 12 port • 35deg spray angle • See Appendix G for details

Figure 2: Photograph of Apparatus

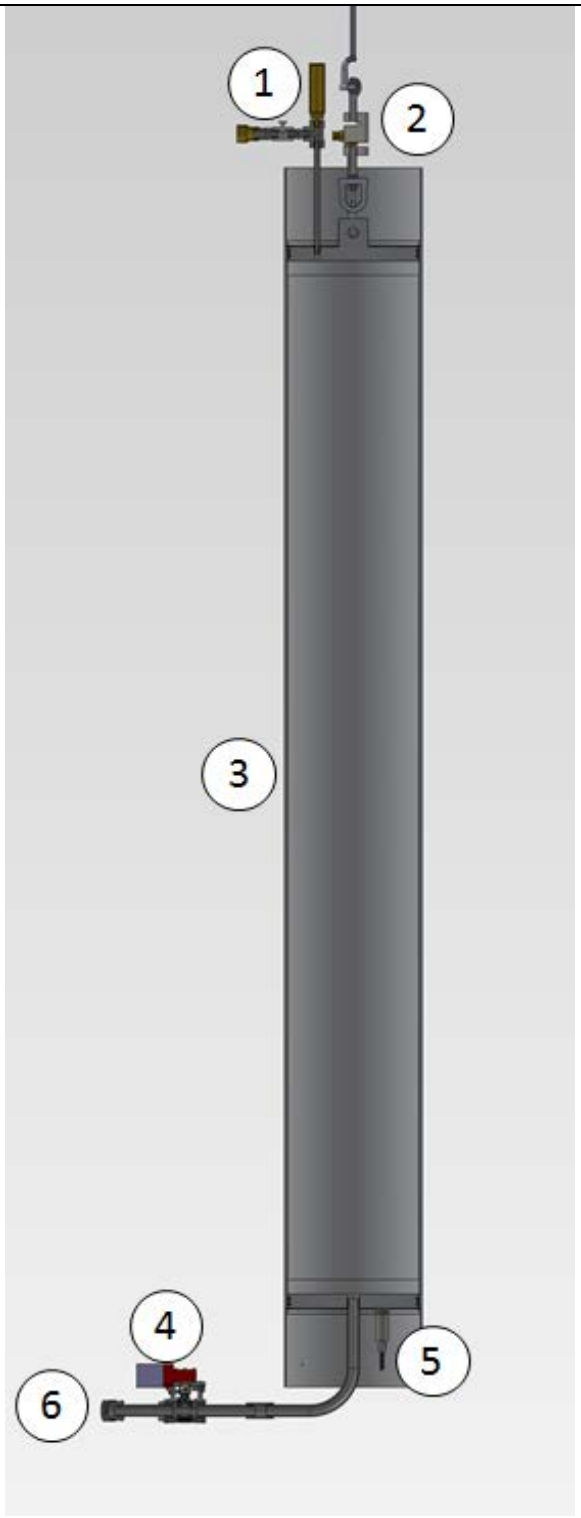


Figure 3: CAD Rendering of Apparatus

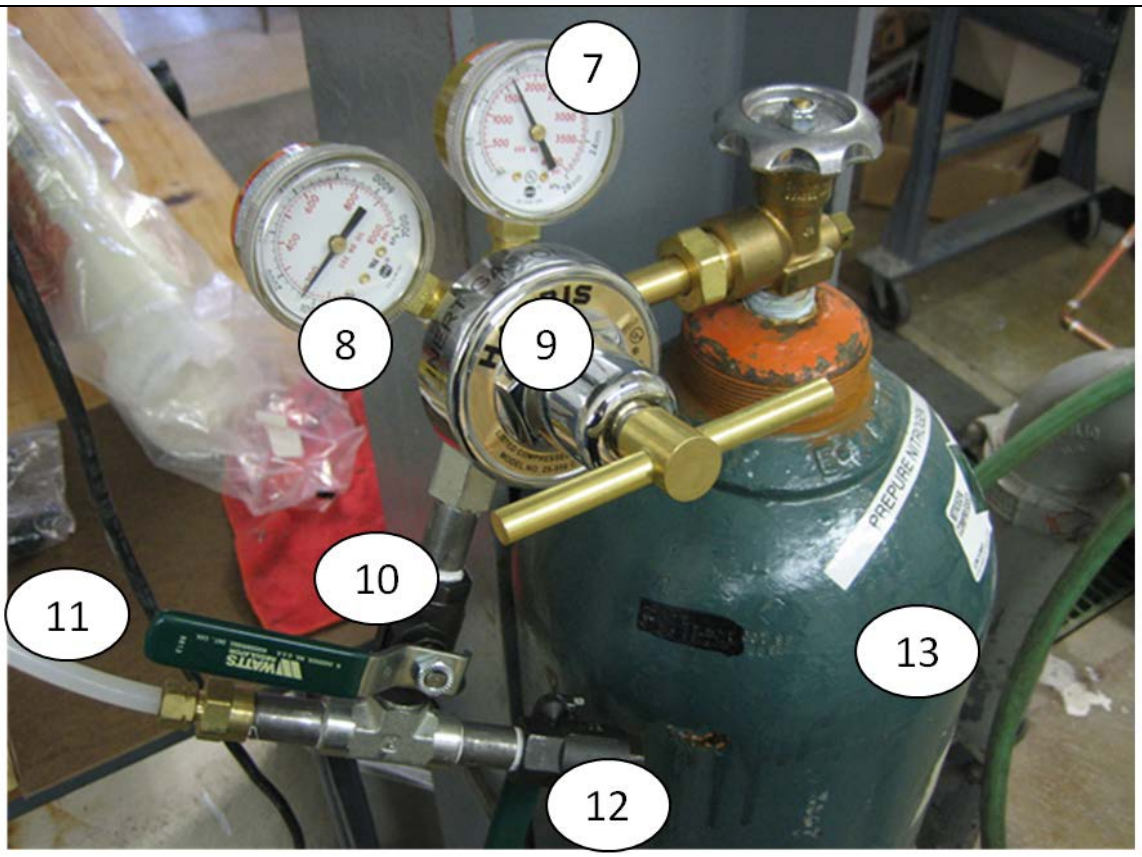


Figure 4: Photograph of Nitrogen Fill Station

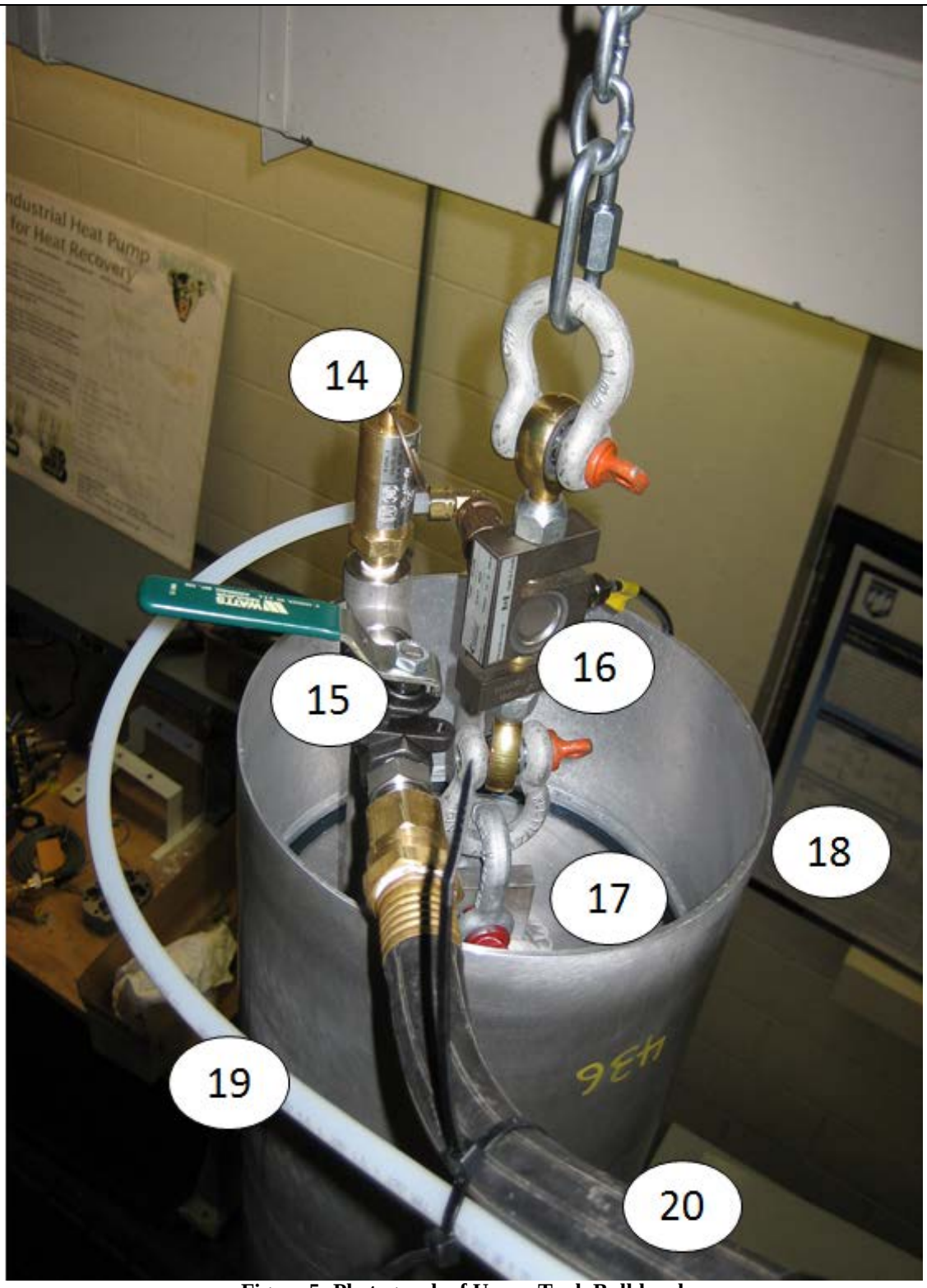


Figure 5: Photograph of Upper Tank Bulkhead

1	Water and nitrogen fill
2	Load Cell

3	Oxidizer Tank
4	Control Valve Assembly
5	Pressure Transducer
6	Injector
7	Nitrogen Bottle Pressure Gauge
8	Nitrogen Feed Pressure Gauge
9	Pressure Regulator
10	Nitrogen Fill Valve
11	Nylon Pressure Hose
12	System Purge Valve
13	Nitrogen Fill Bottle
14	Pressure Relief Valve
15	Water Fill Valve • Identical to nitrogen fill valve and purge valve
16	Load Cell
17	Oxidizer Upper Bulkhead
18	Oxidizer Tank Casing
19	Nylon Pressure Hose
20	Water Fill Hose

3.1.1 Control Valve Assembly

The flow control valve, at least for now, is a ½" stainless steel ball valve. This ball valve is actuated by a NEMA 17 step motor. The step motor does not have the necessary torque to turn the ball valve directly; therefore it needs to be run through a reducing gearbox. The gearbox in question is a right angle worm drive which has a 30:1 gear ratio. To sense when the valve is fully open and fully closed there are two limit switches that are activated by an arm which is attached to the valve stem. The Arduino will keep track of valve position, but when it is initially powered on it needs to find a home position, and to do that it needs the input from a limit switch of some kind. The gearbox was specified to have very low slop, however, the valve stem fits loosely in the valve ball itself; this is accounts for almost all of the system drivetrain slop. It amounts to approximately ½ turn of the step motor shaft before the valve ball begins to rotate. This means there is roughly 6 degrees of slop at the valve. The slop must be accounted

for every time the valve changes direction. Figure 6 is a labeled computer rendering of the valve assembly. Figure 7 is a photograph of the valve assembly. As a reminder, drawings of this assembly can be found in Appendix I.

Table 2: Control Valve Design

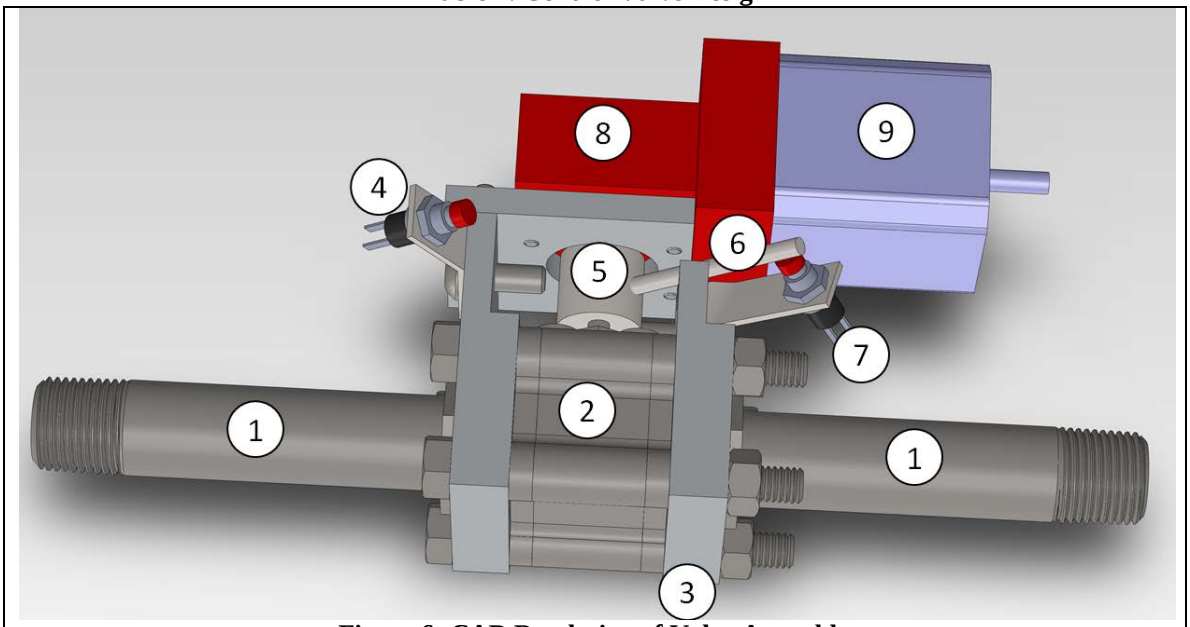


Figure 6: CAD Rendering of Valve Assembly

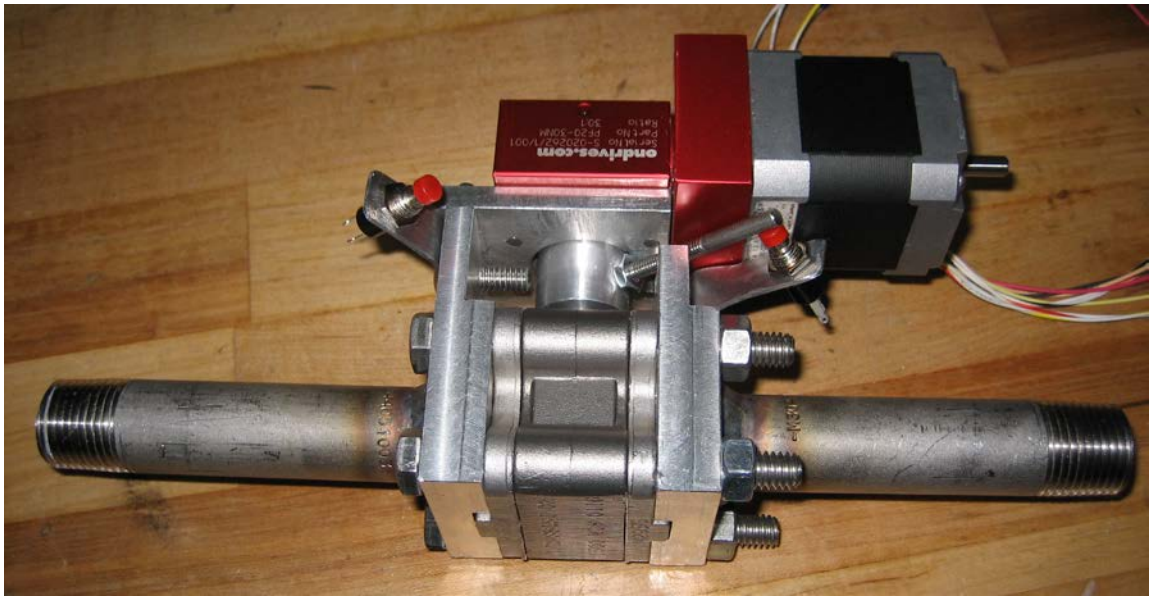


Figure 7: Photograph of Valve Assembly

1	½" SCH 40 stainless steel pipe nipple
2	½" stainless steel ball valve • See Appendix G for details
3	Aluminum motor mount assembly • See Appendix I for drawings
4	Limit switch indicating the valve is open • STSP momentary button switch
5	Coupling from gearbox to valve stem • See Appendix I for drawings
6	Limit switch activator arm • See Appendix I for drawings
7	Limit switch indicating the valve is closed • Identical to item 4
8	Right angle 30:1 worm box • See Appendix G for details
9	NEMA 17 step motor • See Appendix G for details

3.1.2 Oxidizer Tank

The overall tank is machined from extruded aluminum and capped at both ends with flat aluminum bulkheads, held in place with spiral snap rings and sealed with double 3/16" CS Viton O-rings. The tank casing is designed such that it could serve as the

airframe of the rocket booster section. The upper bulkhead has a port for water fill and a second port for nitrogen fill. The lower bulkhead contains a large port for the injector and a second smaller port for a pressure transducer.

The oxidizer tank was designed to fail axially, with a bulkhead failure. Building the tanks from aluminum also reduces the risk of fragmentation if a radial failure was to occur. The tank was designed to withstand 1000 psi, however, in reality; it will be unlikely that it will see anything close to this. All factors of safety are calculated with a tank pressure of 1000 psi. The minimum factor of safety in the axial direction was 1.4, due to bulkhead strength; snap ring groove factors of safety are approximately 2.0. The radial factor of safety is 2.56.

The oxidizer tank casing is machined from 6061 T6 extruded aluminum tubing, 9.5 in OD and .25 in wall thickness. Figure 8 shows a more detailed view of the tank casing end, both ends are identical. Appendix I contains the technical drawings of the tank casing, Figure 8 is simply meant to point out the major design points of the tank casing.

Under pressure, flat plates are not great as tank ends; there was a significant concern with their structural integrity. Therefore an FEA was performed on them to confirm they were up to the job. Figure 9, shows renderings of the upper and lower bulkheads. Appendix I supplies technical drawings. Both bulkheads are machined from 1” thick 6061-T6 aluminum plate.

The snap rings are of the spiral type; the ones used on the oxidizer tank are made by Smalley PN: WH-900. They are rated for 102,130 lbf of radial force, in our particular case the snap ring groove itself will fail before then. The O-rings are made from Viton

with a 3/16” cross-sectional diameter and a dash number of 365. The O-rings were sourced from MSC industrial supply with a PN: 02246064, manufactured by APG.

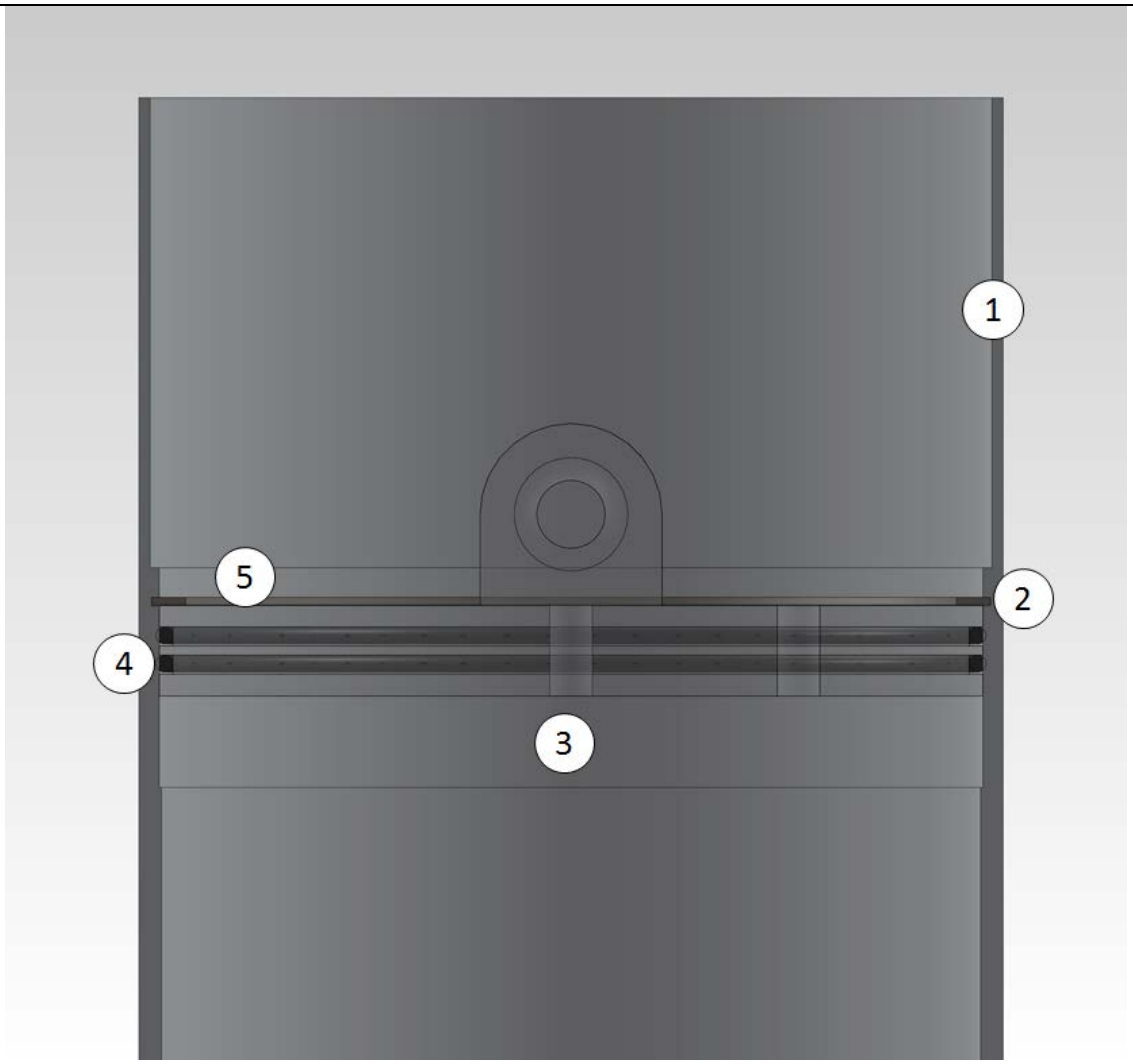


Figure 8: CAD Rendering of Oxidizer Tank End

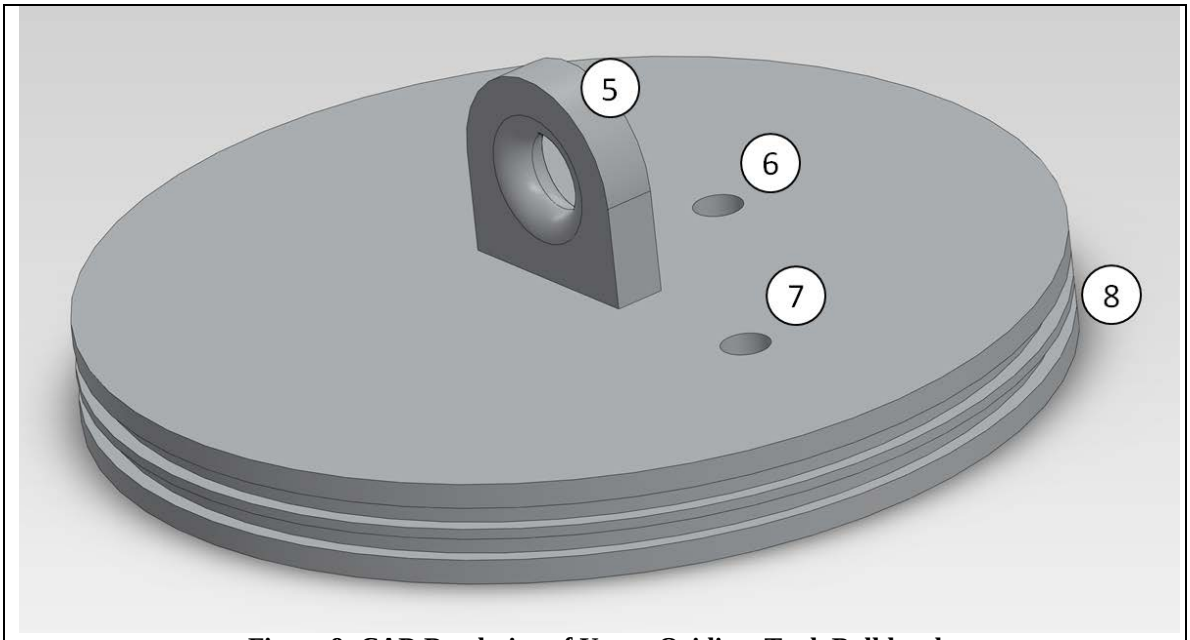


Figure 9: CAD Rendering of Upper Oxidizer Tank Bulkhead

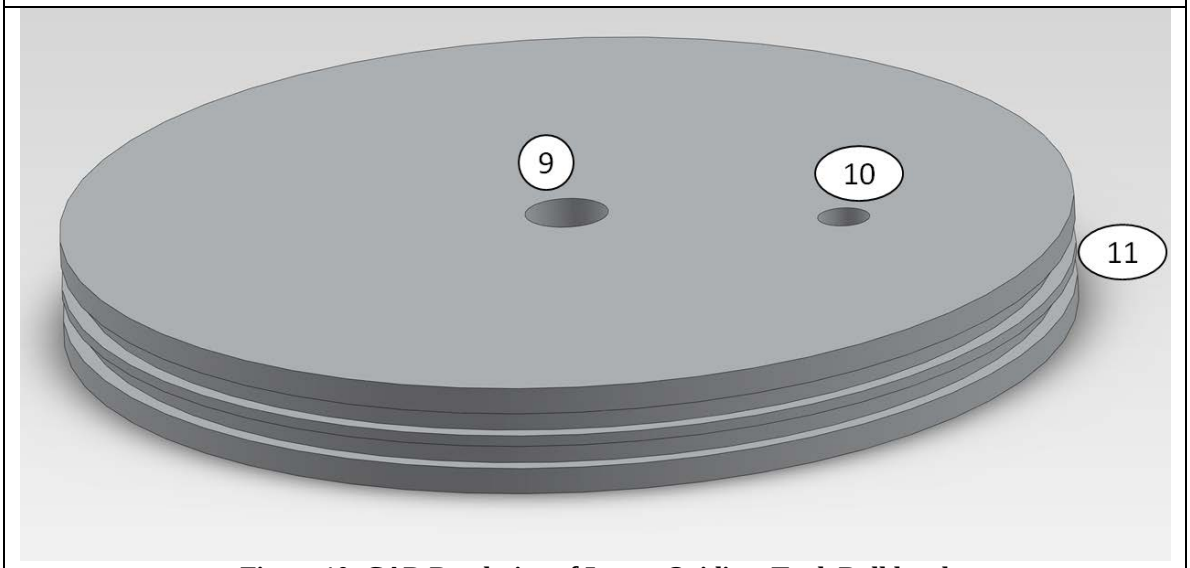


Figure 10: CAD Rendering of Lower Oxidizer Tank Bulkhead

1	Large lap area to join oxidizer tank casing to the rest of the airframe if ever used on a vehicle
2	Snap ring grooves
3	Area machined true on casing ID to provide seat for tank bulkheads
4	O-rings
5	Tab to hang tank
6	Nitrogen fill port (1/4” NPT)
7	Water fill port (1/4” NPT)
8	O-ring glands

9	Oxidizer feed port (1/2” NPT)
10	Pressure transducer port (1/4” NPT)
11	O-ring glands

3.2 Instrumentation and Data Acquisition

The pressure transducer, thermocouple, load cell, and step motor inputs are all recorded on a data acquisition system. The step motor receives input from an Omega Engineering motor controller, which in turn is sent signals from an Arduino microcontroller. The microcontroller handles the control system algorithm and therefore also receives input from the load cell. Figure 11 lays out the wiring of the system.

A word about the motor controller: There are three inputs, one is a step command, one is a direction command, and the last is an enable command. They all operate on a 0 or 5V digital input, and are optically isolated from the rest of the controller. The controller will cause the step motor to move one step (1.8deg) for one on-off cycle of the step command input. The actual step is triggered when the signal drops back to 0V. The direction input defines the direction of the step motor, either 0V or 5V. The physical direction of the motor shaft depends on the motor being used. Lastly the enable command causes the controller to lock the position of the motor and ignore all other input; the motor is locked when the enable pin has 5V and is enabled when the pin is at 0V. This is slightly counter intuitive.

The data acquisition system involves the use of a National Instruments chassis and modules, details are all available in Table 3. 5VDC is supplied to the load cell and pressure transducer, this power source is also monitored by the DAQ. 24VDC is supplied to the Omega Engineering step motor controller, this is a separate supply tailored for the particular motor controller used. Lastly, power for the Arduino is supplied by USB cable

from a laptop. The same laptop is also connected to the DAQ chassis. Figures 12-16 provide photographs of actual components. Remember to refer to Appendix G for specific technical information on any of the major components.

Table 3: Instrumentation Details

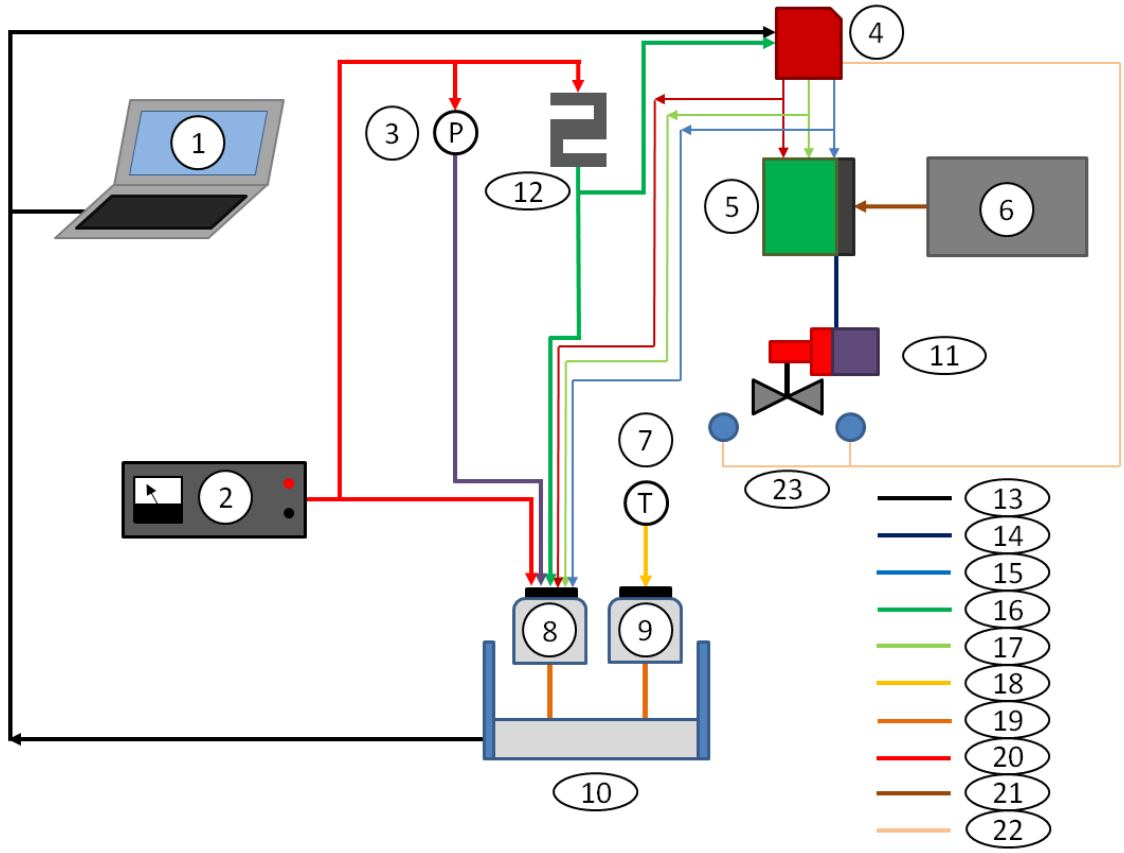


Figure 11: System Instrumentation and Wiring

1	Laptop
2	Sensor power supply
3	Pressure transducer
4	Arduino microcontroller
5	Omega Engineering step motor controller
6	Omega Engineering step motor power supply
7	Thermocouple
8	9205 NI module
9	9219 NI module
10	NI Chassis
11	Step motor control valve
12	Load Cell
13	USB Cable

14	Step motor control cable
15	Motor step command
16	Load cell signal
17	Motor direction command
18	Thermocouple signal
19	NI module to chassis connection
20	5V power
21	24V power for step motor controller
22	Limit Switch Signal
23	Valve Limit Switches

Figure 12: Motor controller

Figure 13: Step motor power supply

Figure 14: Arduino micro controller

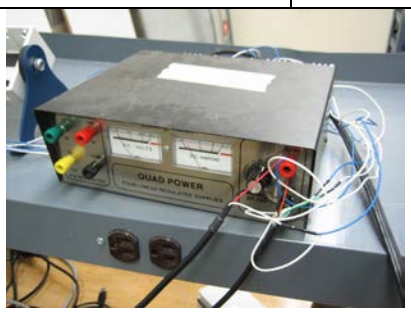


Figure 15: 5VDC power supply

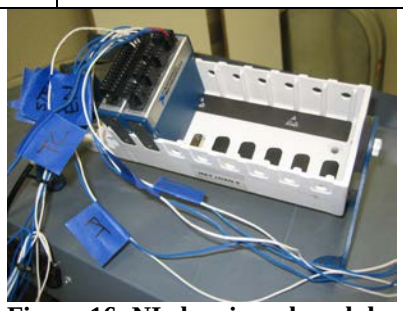


Figure 16: NI chassis and modules

3.3 Description of LabVIEW Program

The physical setup of the DAQ system has already been discussed, there is some setup required on the software side as well. National Instruments distributes the DAQ chassis and modules and also supports a software suit called LabVIEW, which is what was used for this thesis work. LabVIEW, much like MATLAB, uses a visual block and wire method to program the DAS. Figure 17 shows the VI block diagram. Figure 18 shows the program front panel where the user interacts with the program. This front panel is linked to the block diagram; certain function blocks in the diagram create certain user interfaces on the front panel. Since time and expertise were limited, the DAQ Assistant block was used to collect the data, which is essentially a wizard that allows you to choose which DAQ module you wish to use and then walks you through wiring the input to the module correctly. Figure 19 shows how one of the channels within the DAQ Assistant is setup.

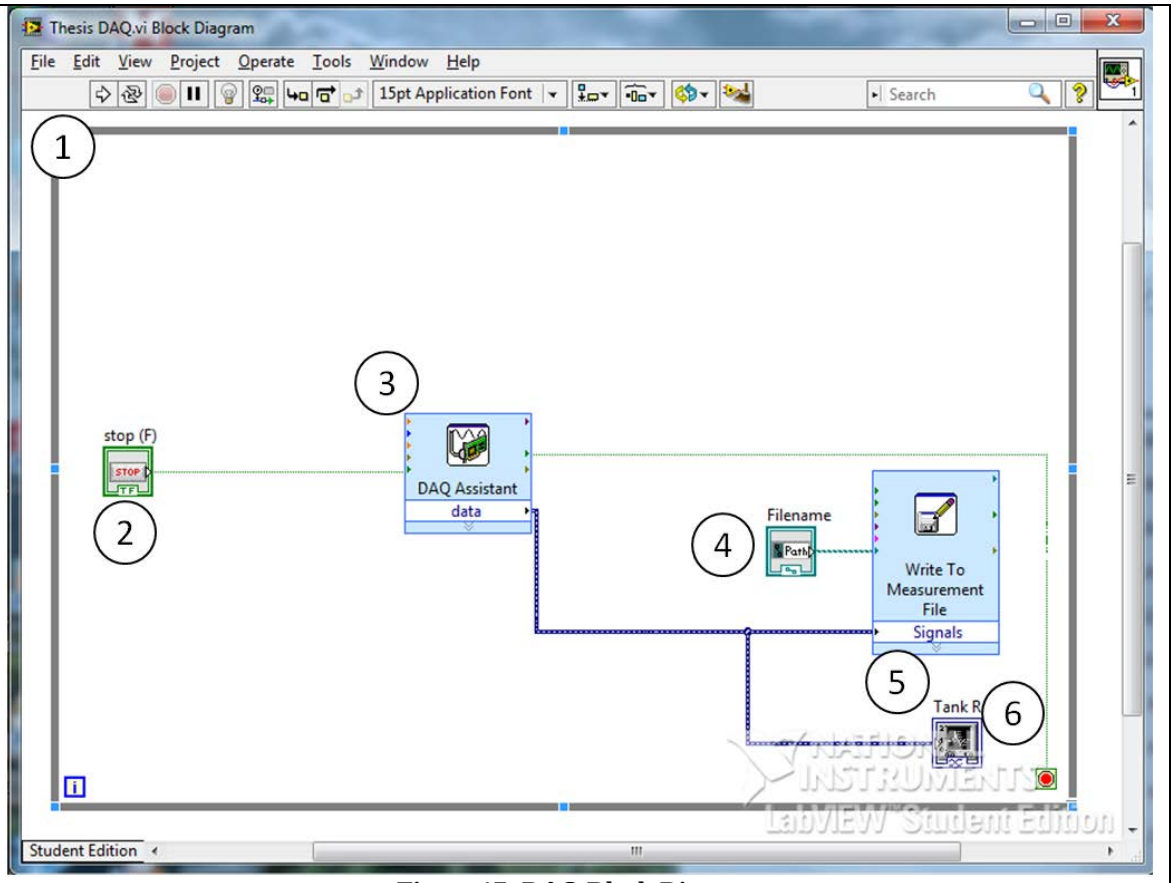


Figure 17: DAQ Block Diagram

1	Loop, causing the DAQ to continuously sample rather than collect a finite number of points
2	Stop button to end data acquisition
3	DAQ Assistant as described below, and shown in Figure 19
4	Filename block used to specify the name of the data file to which the collected data will be stored
5	Write to file block used to store the data to a text file or similar
6	Graph block, allows the user to watch the data in real time as it is being collected.

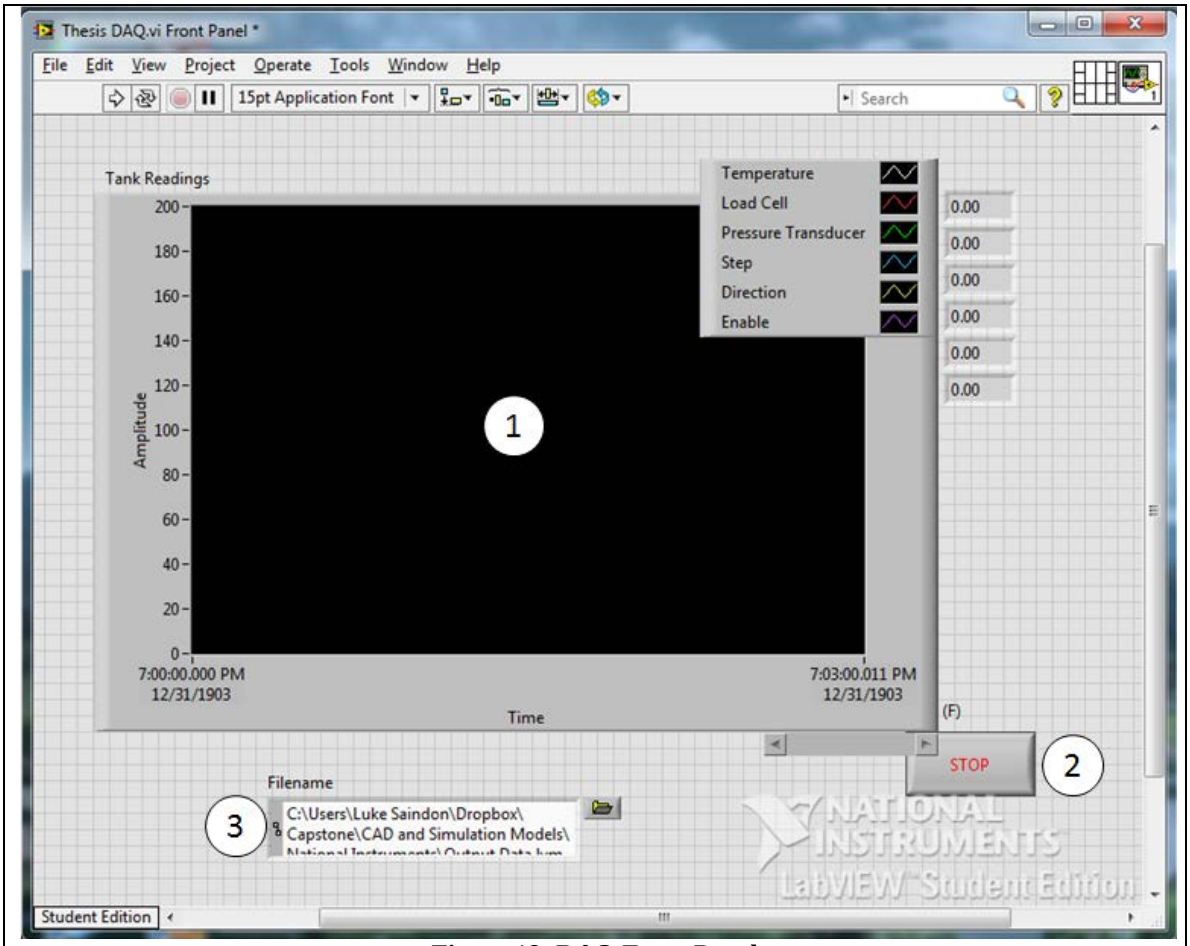


Figure 18: DAQ Front Panel

1	Output graph, created by the graph block in the block diagram
2	Stop button
3	Filename input

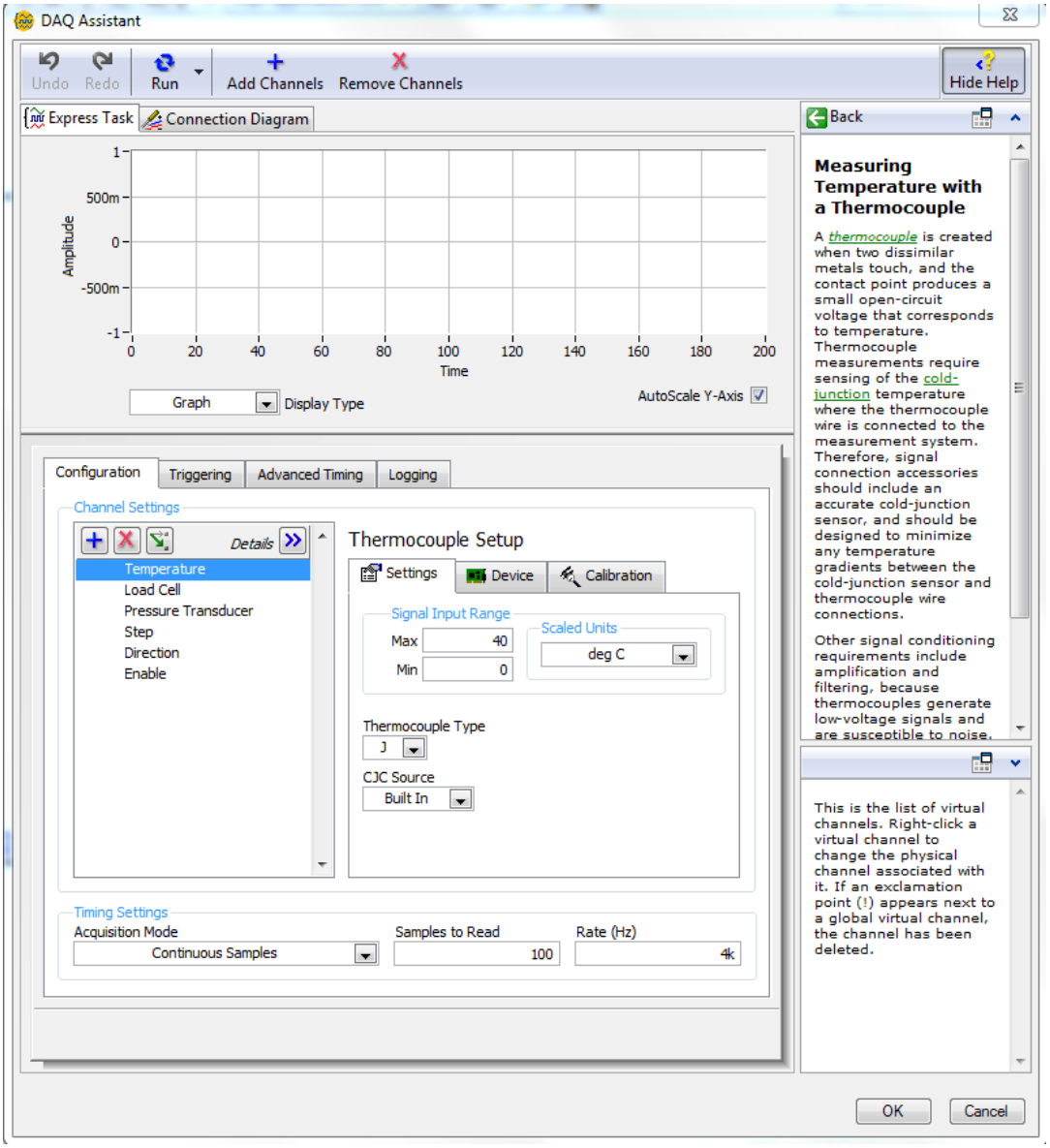


Figure 19: Screenshot of DAQ Assistant Configuration

In Figure 19 the display which sets up the thermocouple is being shown. The wizard allows you to select the min and max temperatures expected to be seen, what units to read the thermocouple in, and very importantly the type of thermocouple. In this case it's a type J. In the process of setting up this channel, the user also has the option to choose which module to use for the data collection, in this case the 9219 was chosen for the thermocouple. The wonderful part of the wizard is the ease at which you can scale

system output. Each of the inputs going to the DAQ is a voltage signal, some in the millivolt range and some in the 5V range. Table 4 lists each of the DAQ inputs in greater detail and exactly how they are scaled, it also lists the order in which they appear in columns when written in the tab delimitated text file. If you have a calibration curve for a sensor it is very simple to configure a calibration function for a specific channel which converts the voltage, current, or resistance signal into another wanted quantity such as weight, temperature, or a variety of other outputs.

Table 4: DAQ Inputs and Outputs

Name	Text Doc Order	Channel	Module	Sensor Output	Scaled Quantity
Time	1	NA	NA	NA	Time (sec)
Thermocouple	2	Ai0	9219	Voltage	Temperature (deg C)
Load Cell	3	Ai1	9219	Voltage	Weight (lbm)
Pressure Transducer	4	Ai2	9219	Voltage	Pressure (psi)
Motor Step Signal	5	1	9205	Voltage	Voltage (V)
Motor Direction Signal	6	2	9205	Voltage	Voltage (V)
Motor Enable Signal	7	3	9205	Voltage	Voltage (V)

3.4 Description of Arduino Program

The program that runs on the Arduino microcontroller is worthy of a more in-depth description; there are several different programs that are currently written for the Arduino which are used throughout the scope of this thesis. One is simply a manual valve control, where you can use a switch to select either open or closed. This is useful for testing the mechanical aspects of the valve or for trouble shooting. The second

program is used for characterizing the valve, and allows a specific valve angle to be specified and loaded into memory. Lastly, there is the controller program which is used to run the closed loop flow control. All the programs are written in Arduino's compiler and can be easily downloaded via a USB link into the chip whenever desired. The code is written in C++, the compiler has many built in functions, however. A screenshot of the program can be seen in Figure 20. All Arduino software is open source.

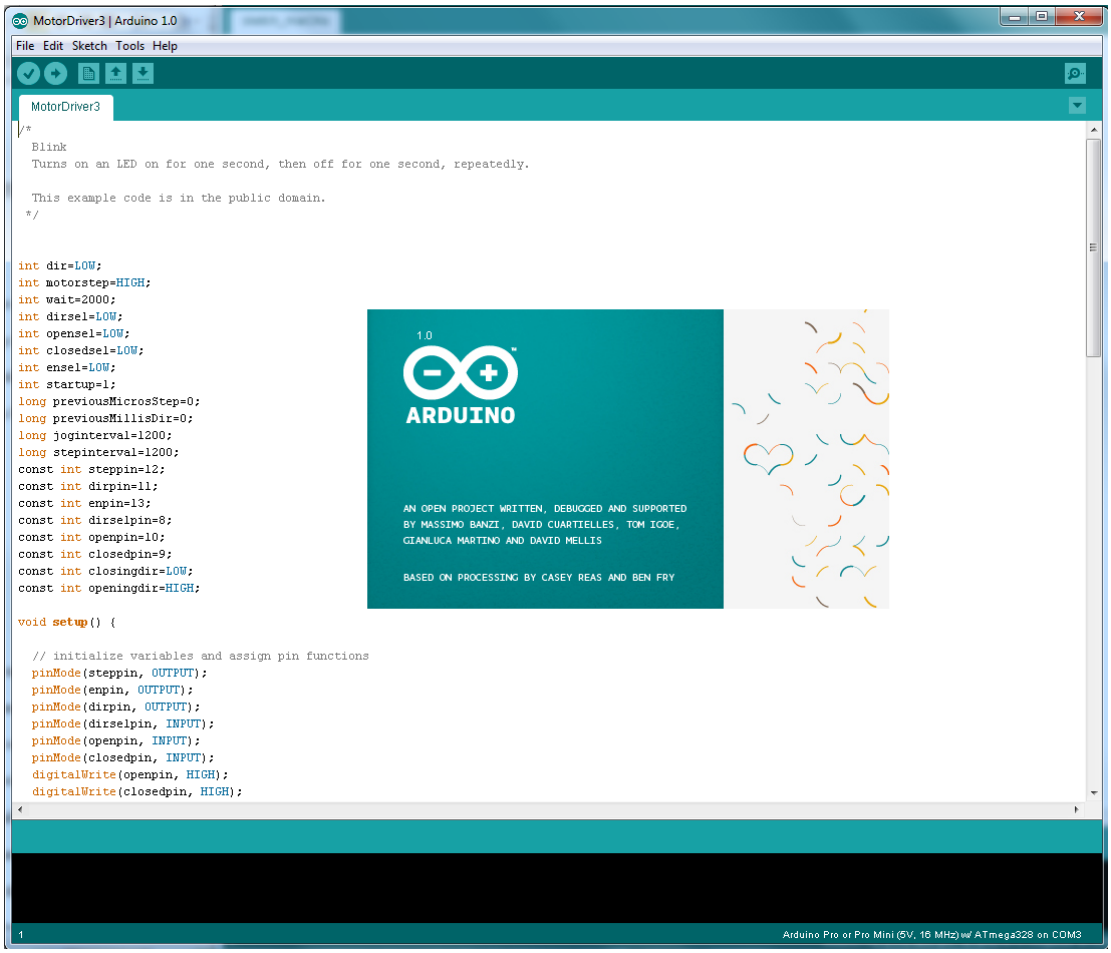


Figure 20: Screenshot of Arduino Compiler

3.4.1 Manual Arduino Program

Again, this program is used to allow manual control of the valve being open or closed, and is the most basic of the three programs. Understanding the way it operates,

however, is the basis to understanding how all the other programs operate as well. A complete printout of the code is available in Appendix D, with an expanded explanation of what is shown below.

At the beginning of the loop the Arduino first reads in the inputs which it receives, such as the high/low values from the valve limit switches, as well as a small manual switch which the user can use to define the direction of motor travel. The next part of code brings the motor back to a home position when it is powered on. The way which the code is setup now is such that "home" is defined as "valve closed". The Arduino knows that the valve has reached the closed position when the limit switch is activated and grounds the associated input pin on the Arduino board. The Arduino does not allow the rest of the code in the "void loop" to be executed until the limit switch indicating a closed position is activated AND the manual direction switch is commanding the motor to move in the closed direction. For example, if when the valve is powered on it is already in the open position and the valve direction switch is commanding the valve to open, the Arduino will cause the motor to shut the valve fully (regardless of user input) and will not allow user control until the direction select switch is also commanding the motor to close.

There is a second pin on the Arduino which gives the motor controller the step command. The step pin oscillates at a rate defined in the initialization portion of the code. The pin is set high or low using a small "if" statement that keeps track how long the step pin is high or low and switches the value after the appropriate interval.

Note that the step pin is oscillating whenever the Arduino has power; the motor is only stopped by using the enable command on the motor controller. For example, when

the valve is fully open and the limit switch pulls the associated Arduino pin to 0V, the code sends a high signal to the enable pin which causes the motor controller to lock the step motor in the current position. All the while the step pin continues to oscillate; its output is simply ignored.

Very importantly there is the code which keeps the valve from going beyond 90 degrees open and 0 degrees closed. This is important because if the valve goes much further beyond these bounds it will bind. To prevent this from happening there are the two limit switches that are activated by the valve stem. As described before, whenever a limit switch is activated it grounds a pin on the Arduino board. If the input pin for either limit switch is activated AND the current direction selection is in the direction of that limit switch a signal is sent to the enable port of the motor controller and the motor is locked in position.

3.4.2 Valve Characterization Program

Generating the necessary information to characterize the valve is a large portion of this thesis work, and requires its own program. As will be explained later in the thesis, characterizing the valve will require repeated runs of the apparatus, all at different valve positions. The valve will be held a constant position for the duration of a flow test, then the tank will be refilled and the valve brought to a different position for a new test. The program is identical to the user controlled program, with the exception of losing user direction control, and an additional condition that causes the motor to be locked in position at the desired angle instead of continuing until it reaches a limit switch.

Appendix E contains a printout of the user controlled Arduino code with additional explanation.

Once the valve has executed the commands that brings it to the home position and the user has flipped the direction control switch to “open”, the Arduino begins to count step cycles with a simple variable that goes up by one every time the step pin completes a cycle; an additional “if” statement compares the step count to the desired angle, and if the step count is greater than the desired angle it locks the motor position.

3.4.5 Closed Loop Flow Control Program

This is the most complex of the programs, and is used to run the closed loop flow control. Unlike the previous two programs, the Arduino will be looking for input from the load cell as well as generating output for the flow control valve. At the heart, the step commands are sent in the same way as the other two programs, except that the step interval is variable, the same initialization sequence to bring the valve to the home position is also implemented. The variable step size allows the angular velocity of the motor to be adjusted beyond either on/off. The program now also receives an analog input from the load cell. The Arduino then differentiates this mass measurement to get a mass flow reading which can be compared to the reference signal. Depending on the error, an appropriate valve angular velocity is determined, which is then converted into a given step interval. Since the motor controller has a minimum step interval that it can recognize there are saturation limits placed to constrain how small the step interval can be made. Appendix F contains a copy of the code along with a much more detailed description.

This desired angular valve angular velocity must be translated into the mentioned step interval for the motor, if the input is small (ie a low angular velocity) then the step interval must be longer (ie a larger pause between step commands), higher angular velocity means a short step interval. The derivation of the equation that converts a valve angular velocity to a step interval is shown below.

$$\dot{\phi} = Motor\_RPM = \frac{1.8\left(\frac{\deg}{step}\right)}{step\,\mathrm{int}(\sec)*2}$$

$$\dot{\theta} = \frac{\dot{\phi}}{30\left(\frac{\deg}{\deg}\right)} = \frac{1.8\left(\frac{\deg}{step}\right)}{step\,\mathrm{int}(\sec)*2*30\left(\frac{\deg}{\deg}\right)}$$

$$also:$$

$$\dot{\theta} = KP*error + KI*\int error*dt$$

$$\therefore$$

$$KP*error + KI*\int error*dt = \frac{1.8\left(\frac{\deg}{step}\right)}{step\,\mathrm{int}(\sec)*2*30\left(\frac{\deg}{\deg}\right)}$$

$$step\,\mathrm{int}(\sec) = \frac{1.8\left(\frac{\deg}{step}\right)}{\left(KP*error + KI*\int error*dt\right)*2*30\left(\frac{\deg}{\deg}\right)}$$

$$step\,\mathrm{int}(\mu\sec) = 1*10^{6}\,\frac{1.8\left(\frac{\deg}{step}\right)}{\left(KP*error + KI*\int error*dt\right)*2*30\left(\frac{\deg}{\deg}\right)}$$

Equation 3: Determining Step Interval from Valve Angular Velocity

Lastly, there are two "if" statements that determine the direction of the motor depending on whether the "input" (ie valve angular velocity) is positive or negative. The limit switch protection is also still in place to prevent the valve from binding.

CHAPTER 4: MODELING THE SYSTEM

Once the valve is characterized the results will be used in a computer model of the oxidizer flow system. The hope is that an accurate model of the system can be made on the computer such that tests can be run in MATLAB instead on the actual test rig, which will allow more rapid tuning of the control system as well as provide a characterized component (the oxidizer feed system) to a simulation of an entire motor setup when the time for that arrives.

4.1 Assumptions about the system

In order to model the system it was necessary to make several assumptions about the way the oxidizer behaves as a fluid, how the control valve operates, and the blow down process that occurs as the oxidizer leaves the tank. The assumptions made during this investigation will now be presented:

Incompressible fluid, since water will be used it was assumed that the fluid is incompressible, therefore no cavitation at the injector or the control valve was accounted for. It also means that any choking effects at the injector were ignored. This assumption may prove to be the most problematic when transitioning to the real oxidizer, which will most likely be nitrous oxide. Nitrous oxide behaves as an incompressible fluid, while it remains a fluid. However, under the pressures and temperatures we will be working with the nitrous is very close to the saturated vapor phase. This means that any pressure drops through the control valve could very easily result in momentary vaporization of the fluid, resulting in compressible gas pockets. These gas pockets (cavitation) may severely choke the flow if they occur, this will have to be investigated in detail once the system is

characterized using a fluid other than water. Carbon dioxide has very similar properties to nitrous oxide and would serve as a cheaper and safer analog and therefore a stepping stone to nitrous. Due to time limits carbon dioxide will not be tested in the scope of this thesis. The valve, injector, and controller can be characterized in the same way as with water; the results may be very different. The concern is a drastic drop in system flow capacity when using fluids near their vapor pressure.

Fluid temperature, the temperature of the fluid is considered the constant room value. The fluid temperature will be monitored just before it enters the control valve, but there are no compensations for that temperature measurement in the simulations. Since the fluid is also considered incompressible it follows that the density of the fluid is then also considered constant when using Bernoulli's equation and the energy equations involved with the fluid dynamics of the system. Large swings in temperature have not been observed and therefore this is deemed a reasonable approximation.

To simulate the system accurately the pressure in the oxidizer tank needs to be known so that the pressure drop across the flow path can be determined. This pressure drop, of course, is the key component to understanding and predicting the mass flow of fluid. As will be explained in more detail later in the report, the ullage space within the tank will be supercharged with nitrogen and the system will blow down to atmospheric pressure. Modeling this blow down process involved making several assumptions about the behavior of the nitrogen. Firstly, it was considered to be an ideal gas. The compressibility factor never drops below approximately .95. As detailed later this also involved assumptions about the ideal gas constant used. Secondly, only the nitrogen constituent of the ullage gas was considered. In reality there will be a mixture of water

vapor and nitrogen. After each run, the tank will be completely flooded with nitrogen, which will reduce the buildup of water vapor in proportion to the nitrogen. The biggest assumption, which may warrant further investigation, is that the process is considered adiabatic, in other words there is no heat transfer between the tank and the surrounding environment. The blow down durations are relatively short, roughly 30 seconds, therefore this assumption seemed reasonable as a first pass.

4.2 Governing equations

Now that the general layout of the system is presented the operation of each of the components can be explored in more detail. Then the simple equation used in the simulation will be presented. First of all the control valve itself will be examined.

The control valve is of the ball type; therefore it operates from open to close over a 90 degree rotation. Unfortunately the loss coefficients through the valve do not vary linearly from these two extremes. Regardless of whether the flow response is directly proportional to the valve position or not, one of the variables that significantly affect oxidizer flow is valve position. The second significant factor is the pressure drop which is present across the flow path. In this simulation, the flow through the control valve was limited to two factors, the valve position, and the pressure drop. A variable, called (coined for this experiment) the valve factor, or VF, was defined such that a surface of VF plotted against valve position and pressure drop could be created. Using this new variable and the equation for mass flow of an incompressible fluid from Bernoulli's equation the approximate mass flow response of the system can be put in the form of Equation 7. A loss coefficient was added. The streamline used is shown in Figure 21.

$$\dot{m} = \sqrt{\frac{2\rho\Delta P}{D_{loss}}} = L \times VF \times \sqrt{\rho}$$
$$VF = VF(\Delta P, \theta)$$
$$L = \sqrt{\frac{2}{D_{loss}}}$$

Equation 4: Mass flow based on pressure drop and valve position

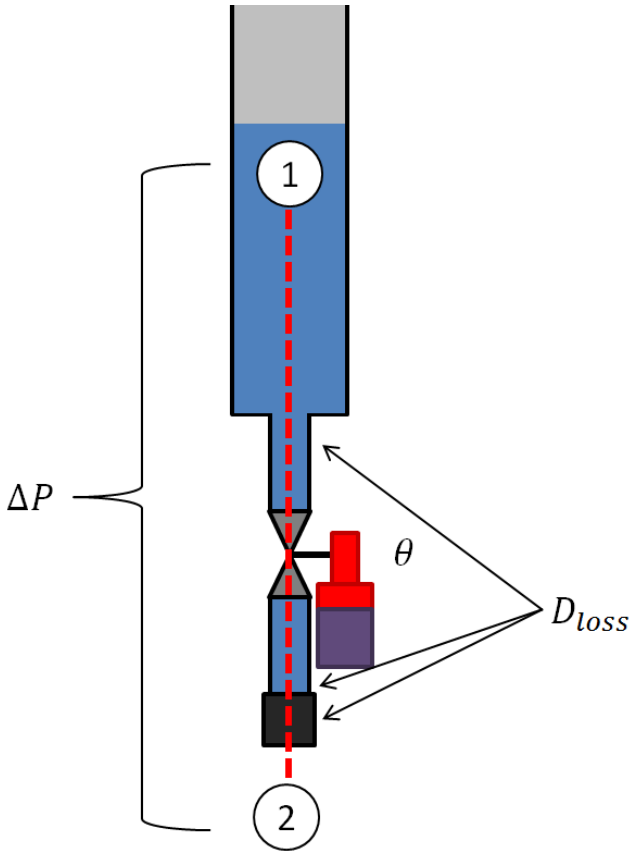


Figure 21: Bernoulli's Streamline

The second version of Equation 7 shows a variable named “VF”, VF is the above mentioned “Valve Factor”. VF varies between 0 and 1, depending on the valve position and pressure drop. 0 would represent no flow, and 1 would represent maximum system flow. The max flow was based on achievable upper bounds. The trend between the 0 and 1 for a constant pressure drop and varying valve position is not linear. VF also lumps

the mass flow's dependency on the square root of pressure. A figure of the VF surface is presented in the results section of this report, figure 32. The Valve Factor takes into account the trends in the system as the valve opens and closes and pressure falls, but does not translate valve position and pressure drop into an actual mass flow number. There are also other losses in the system, such as the injector, and the tubing. This residual resistance of the system as well as the constant which multiplies VF to give actual mass flow numbers is all contained in the "L" coefficient, which is called the "VF multiplier". The VF multiplier (L) is chosen such that when the valve is fully open the predicted mass flow is appropriate for a given pressure drop across the tubing, injector, and open valve. Once again, this means that VF serves only as a proportional term to correct for valve position and pressure drop. Constant parasitic losses in the system are accounted for in the VF multiplier (L). Finding good values of VF as a function of valve position and pressure drop is one of the primary and most necessary objectives of having a working simulation the system. Note that the square root of density is left as a separate term, but since one of the system assumptions was that of an incompressible fluid this will remain effectively as a third constant.

Initial values of L and VF were estimated for the first pass using the data sheets from the injector manufacturer and valve manufacturer as well as a Darcy friction factor calculation for losses in the tubing. Note: an arbitrary pressure drop had to be assigned the valve factor value of 1 (ie the maximum expected pressured drop to be seen), in this case 85 psi of pressure drop. Of course the maximum flow will be with the valve wide open, 90 deg. All other pressure drop/valve positions were scaled from this position; you can see that the back corner of Figure 32 is the position chosen for VF to be 1.

4.3 Modeling the valve response (effects of delay)

In reality the response of the valve to the input of the controller is somewhat flawed. The main component of this, and the only thing accounted for in the simulation, is the slop within the mechanism. The step motor that controls the valve is severely geared down; therefore any slop in the gearing is amplified by the time it reaches the motor shaft. The motor has roughly 180 degrees of free rotation before the slop is taken up at the valve. This is modeled by adding a backlash block in the Simulink model.

4.4 MATLAB simulation description and explanation

The entire system simulation is performed within MATLAB, within MATLAB there is a package called Simulink which specifically designed for use with control systems. All computations are carried out in a Simulink block model, but all of the variables within the code are initialized using an .m file, the actual Simulink model is also run from the .m file using the "Sim" command. This way all that needs to be done to run a simulation is open the .m file and execute, the Simulink model will be run in the background without being touched by the user. Data from the model is saved to the workspace and then plotted by the m. file, displaying the reference curve and the simulated response superimposed. Once actual mass flow curves are experimentally found a third "measured" mass flow curve can be added to the plot.

Simulink uses a block model flowchart layout, making the design of a simulation fairly easy and intuitive. Blocks are connected with wires to control the flow of calculations. Blocks can be as simple as an arithmetic operation, or they can be a complicated string of embedded code that the user writes. Usually a group of operations

are consolidated into a subsystem, such that overall the model looks cleaner and is easier to follow. Figure 21 shows the top level Simulink model used. A breakdown of the entire Simulink model can be found in Appendix A.

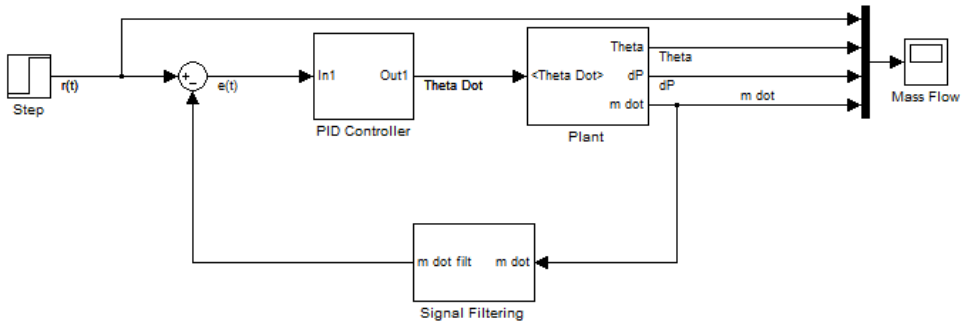


Figure 22: Overall Block Diagram

The block labeled "Step" is the reference curve. This is the curve that the controller will be attempting to force the system output towards. In this example, it is a simple step input, however, this block can be modified to be any desired signal shape. The feedback loop, which was discussed earlier, joins this reference curve in a simple addition/subtraction block. This results in the error signal which is fed into the controller subsystem. Figure 22 displays the controller subsystem block.

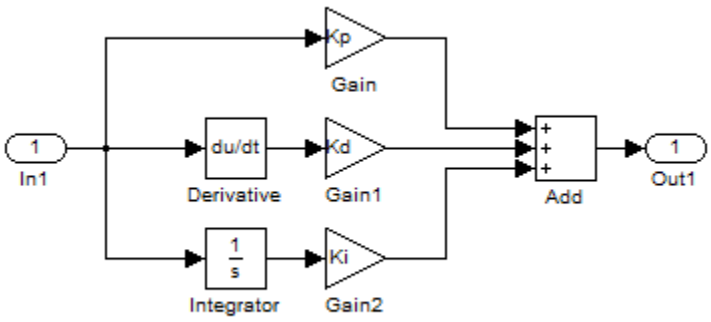


Figure 23: Controller Block Diagram

The controller subsystem has the capability to be a complete PID controller, even if the full functionality may not be used. The error signal is feed into three branches, one for the proportional controller, one for the integral controller, and one for the derivative controller. The signal is then differentiated, integrated, or left alone before it is multiplied by the appropriate coefficients. The three signals are then summed before being used as the controller output/system input. If only part of the controller functionality is desired the appropriate controller constant can be set to zero in the program initialization sequence (the .m file). The output of the controller is a command for the angular velocity of the valve (a direction and rotational speed). In the actual apparatus this will be translated into a "stepinterval" for the motor controller.

Now that the flow has passed through the controller subsystem the simulation must now predict how the system will respond to this controller output. This is the purpose of the "plant" block. This is where the angular velocity input is translated into a new valve position and ultimately a mass flow through the oxidizer system. Figure 23 shows the plant subsystem.

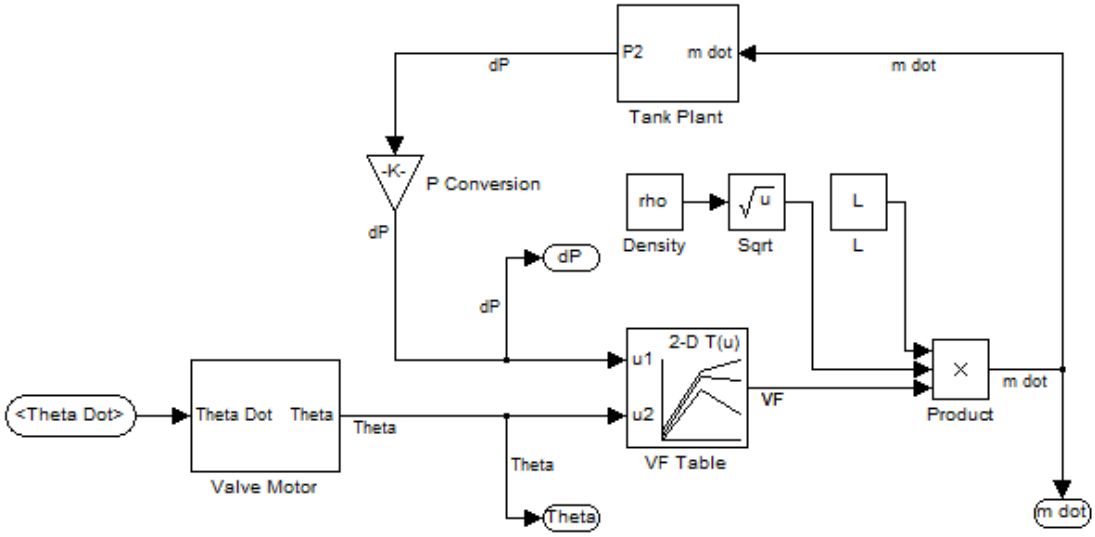


Figure 24: Simulation Plant Subsystem

The first block that the input signal goes through is the "valve motor", this is where the valve angular velocity signal is translated into valve position. A diagram of this block can be found in Appendix A. Essentially what happens is the angular velocity is fed through an integral block which outputs angular position (theta). The time delay involved with drivetrain slop is also handled here with a backlash block. Lastly, the controller may specify a very high angular velocity. However, the motor can only provide a certain speed; often not as high as the controller would like. This is called saturation, and the simulation needs to account for it. In this "valve motor" block there are limits that filter the controller signal to truncate it when it reaches these mins and maxes. The controller also does not understand the idea that the valve may already be completely open or closed; if the mass flow is lower than it wants but the valve is actually already fully open the controller will still attempt to open it further. Therefore, there needs to be a second set of saturation parameters that keep the valve from opening

any further or closing any further than it physically can. This is similar to the limit switches on the actual apparatus.

The VF table block is really the heart of the simulation, and perfecting its contents is an important goal. This block is a lookup table that contains the value of the valve factor (VF). There are two inputs to this block, theta (valve position) and pressure drop across the oxidizer flow path. From these two parameters the value of VF can be looked up. A 3D surface plot of this lookup table can be seen in the results section, Figure 32. During the valve characterization progressed, the contents of this lookup table was fine-tuned such that simulation matched results.

Now that a mass flow has been calculated given the angular velocity output of the controller, the last major simulation step in the plant is to calculate the behavior of the fluid in the oxidizer tank. This step is needed since a new pressure drop will be needed in the next simulation loop. A diagram of the tank plant subsystem is shown in figure 24.

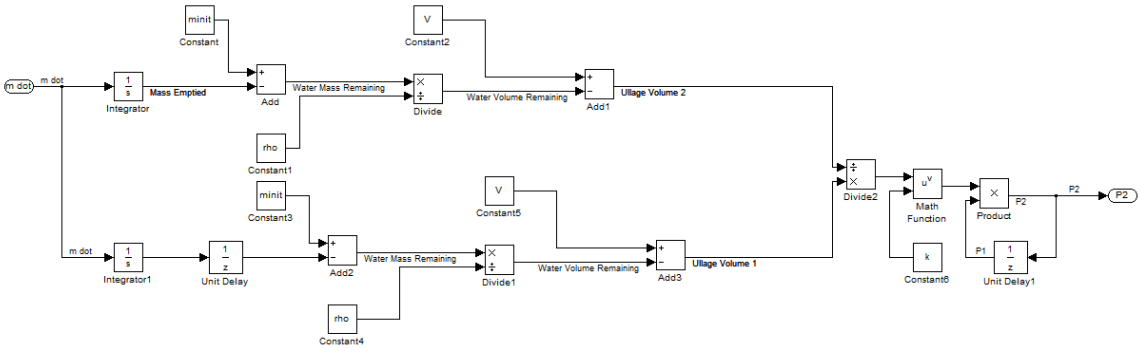


Figure 25: Tank Simulation Subsystem

It works by finding the change in volume of the ullage gas, this can be done because the time step of the simulation and the instantaneous mass flow are known. If a constant mass flow for that small amount of time is assumed then the change in mass of the tank

contents can be found. The water is assumed to be incompressible, therefore a change in volume can be found. Using this change in volume and the assumption that the system is adiabatic the final pressure can be found. The initial pressure is simply the pressure from the previous time step. The pressure drop across the control valve is simply calculated by subtracting atmospheric pressure from this newly found tank pressure. Also, as pressure drops, more water vapor will enter the ullage gasses. However, this error was small and thus ignored. The ullage is considered to be pure nitrogen for the duration of the oxidizer tank blow down. In Figure 24 the upper branch is the calculation of ullage volume at the current time step, whereas the lower branch has a unit delay block and represents the ullage volume at the previous step. Using these volumes as initial and final volumes, knowing the initial pressure, and the ratio of specific heats for nitrogen the final pressure is found. This means the simulation loop is primed for iteration and a new error value is found. The process is repeated.

CHAPTER 5: TESTING PROCEDURE

There are two main operations that this system is designed for; one is to create the VF plot that is necessary for creating an accurate system model using this particular ball valve. The other is for testing the performance of the flow control system once the valve is characterized. The procedures associated with each of these operations are slightly different, however, share many similarities.

5.1 Characterizing the Valve

First, the methods of creating a VF plot will be reviewed, including the procedures for starting the apparatus from a cold start, running a single test, resetting for another test, and then shutting down the system. The valve characterization method used here is to run flow tests at constant valve position, while allowing the tank to blow down from an initial pressure to atmospheric pressure. To get a complete VF surface it is recommended that the apparatus be pressurized to 150 psi initially and then run with a constant valve position; this should be repeated for valve positions from 0 to 90deg in 5deg increments.

5.1.1 System Preparation from a Cold Start

1. Power on Arduino by plugging the USB cable into the board and then into a USB port of a computer
2. Ensure that the “Specific Angle.ino” is loaded onto the Arduino by opening the compiler and re-downloading. Before downloading be sure to enter the desired angle into the program.

3. Turn on the step motor power supply and allow the program to close the control valve if it is not already closed.
4. Using the direction select switch, cycle the valve open closed several times to ensure that nothing is sticking or binding. If the valve has been sitting for an extended period of time it may be a bit sticky at first, causing the motor to falter. This is also an opportunity to make sure the tank is empty. When done leave the valve in the closed position.
5. Open the VI, power on the sensor power supply, and turn on the NI chassis
6. Run the DAQ program and watch the load cell readout to ensure it is zero
7. Check that valve 2, 6, and 12 in Figure 1 are closed, also ensure the regulator is shutoff.
8. Check that valve 10 is open.
9. Open valve 12 to begin filling the tank until the load cell reads 172lbm, this will leave the tank 91% full; the perfect amount if it will be pressurized to 150psi with nitrogen, since it will leave 0psi by the time the tank is empty. This allows a full range of pressure readings to be attained from a given valve position. If less pressure is desired initially then the tank should not be filled as much, and vise versa.
10. Shut valve 12 when the tank is at the desired level. If the tank is accidently overfilled then command the flow valve to be opened briefly and allow water to partially drain, and then try again.
11. Close valve 10
12. Open valve 2

13. Adjust regulator to 150 psi
14. Open valve 6 and allow the system to reach equilibrium
15. Check the DAQ to ensure the pressure reading is at 150psi, adjust the regulator if necessary
16. Close valve 6
17. Stop the DAQ and discard of the old data file that was created while preparing the system. The system is now filled with water, pressurized, and ready for the flow valve to be opened to collect data.

5.1.2 Running a Test

1. Execute only after performing the system preparation
2. Enter a filename and file path for the data file to be stored
3. Run the DAQ VI
4. Flip the motor direction selection switch and watch the valve move to the entered angle
5. Collect data until the load cell or pressure transducer reads 0.
6. Flip the direction select switch to close the control valve

5.1.3 Resetting for another Test

1. Change the desired angle in the Arduino compiler and re-download to the Arduino board
2. Open valve 10
3. Run the DAQ VI

4. Execute steps 9-11 of “system preparation”
5. Open valve 6 and allow the system to equalize
6. Execute steps 16 and 17 of the “system preparation”
7. The system is now ready to re-run a test

5.1.4 System Shutdown

1. Close all valves
2. Open valve 10
3. Open valve 6, allow the regulator to depressurize
4. Close regulator
5. Shutdown the motor power supply
6. Turn off NI chassis
7. Unplug the Arduino from the laptop
8. The system is shutdown

5.2 Testing the Control System

In the process of tuning the control system and fine tuning the system simulation it is necessary to test the response of the system to controller, and then compare to the computer model. The hope is that once a good computer model is implemented, actual tests of the control system will be fairly limited. Much of the system initialization is similar between testing the control system and characterizing the valve.

5.2.1 System Preparation from a Cold Start

This is the same as the system preparation for characterizing the valve, except in step 2 load the “controller.ino” program into the Arduino’s memory. Remember to set the desired reference curve before downloading the program into the Arduino.

5.2.2 Running a Test

This procedure is also the same as the instructions for characterizing the valve, except expect the valve to be constantly moving while it attempts to follow the defined reference curve.

5.2.3 Resetting for another Test

Also the same as for valve characterization, however, re-load the Arduino program with new controller constants instead of a new angle.

5.2.4 System Shutdown

This procedure is the same as that for the valve characterization.

CHAPTER 6: POST PROCESSING

Some work, such as scaling the raw sensor output into quantities like weight, pressure, or temperature instead of pure voltage values has already been performed in the DAQ program, as explained in the apparatus description; but there is a still lot of analysis and manipulation that is performed after the fact in MATLAB. The operation and use of the post processing MATLAB programs will now be explained, there are two different programs. One constructs the VF plot, and the other analyzes the performance of the control system.

6.1 VF Plot Post Processing

When the valve is held at a constant position while the water drains from the tank the DAQ system is recording the mass of water over time, therefore to find the mass flow over time the mass vs. time data must be numerically differentiated. One of the largest challenges is getting a clean plot of the mass flow curve since there is some high frequency noise in the mass curve. This difficulty is demonstrated with the very simple example shown in Equation 2. The solution found for this was to use a polynomial curve fit to get a clean line for the mass vs. time curve and mass flow vs. time. MATLAB has a built in polynomial fitting tool, which was used after passing the data through a Savitzky-Golay filter to slightly reduce the noise before fitting the curve. Appendix B provides a copy of the MATLAB code used.

The data from each of the test runs is stored in a .lvm file, there is an individual file for each valve position between 0 and 90deg in 5deg increments. Smaller increments can be taken, depending on how accurate you want the plot to be and the amount of time

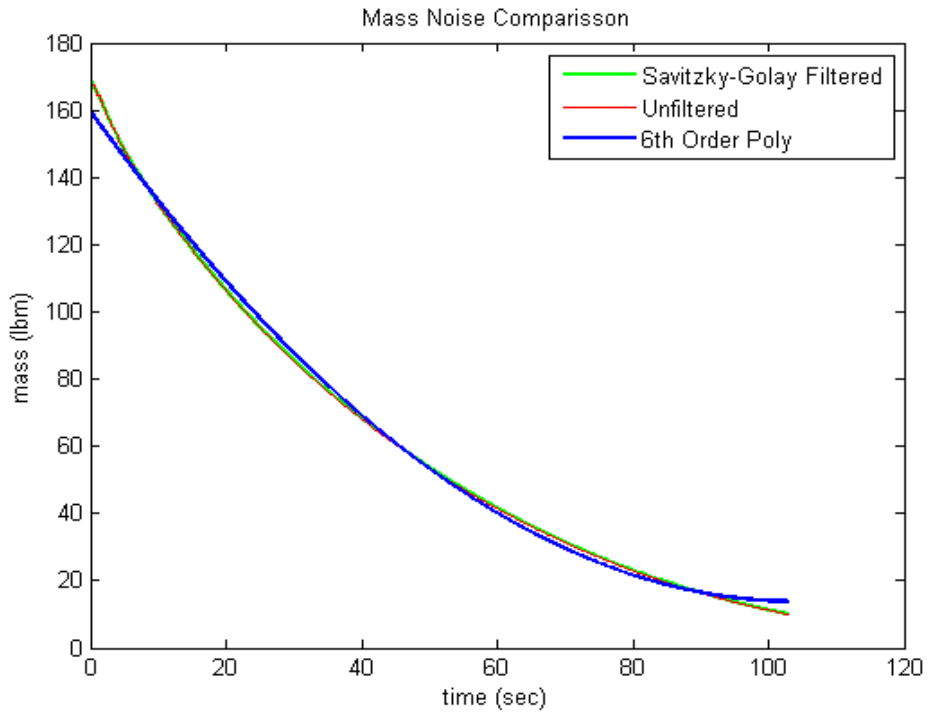
available to run the tests. In order for the MATLAB program to work correctly, each of the files must have a unique name and they must all be in the same directory along with the .m file. The first step the program needs to take is to read the data from the text files and store it within its own variable.

Two operations are executed immediately; first the Savitzky-Golay filter is run on the temperature, mass, and pressure data. Neither the pressure or temperature data is at all noisy, and the mass data is quite good as well, however, the 9219 cannot sample as quickly as the 9205 yet the NI chassis clock is set to sample at a high enough rate that the 9205 can sense the step signal from the Arduino. The step signal can be running at a frequency as high as 500Hz, therefore the DAS records the same values from the 9219 for several consecutive time steps. This creates a stair step signal from all the 9219 sensors (load cell, thermocouple, and pressure transducer). The Savitzky-Golay filter is very effective at linearizing this data without distorting the amplitudes.

The next step is to differentiate the mass flow vs. time curve. This is done with a simple Euler method, find the difference between consecutive mass values and then divide by the elapsed time between the two samples. The data which is differentiated is the Savitzky-Golay filtered mass curve, not the unfiltered mass vs. time curve. Also, once the curve is differentiated to find mass flow vs. time the curve is run through the filter again. To get a very clean line for the VF plot this filtered mass flow curve is then fitted with a 6^{th} order polynomial. A second order fits reasonably well, but the 6^{th} order allows the curve to be a much more "blunt" parabola, which fits better. Figures 25-31 show sample data before any filtering has been done, and then have an overlay of the data once the filter has been applied, and then once again when a polynomial curve fit is used.

The R^2 values for the curve fits are displayed beneath the charts. Inserts showing the details of some noise are also shown. Note that the polynomial curve fit for the mass vs. time curve is actually not used in any calculations, whereas the polynomial curve fit on the mass flow vs. time curve is used in constructing the VF table.

The last process which occurs is to crop the data which was collected. There is a significant amount of data at the beginning and the end of the test which involves dead time before the valve is open or after it is closed as well as transient regions before the valve has reached its final position. The user must look at the raw data from each test and determine time bounds for the data. These time bounds are entered into the processing program and all data that doesn't fall between the start and end times is discarded.



R^2 polynomial/unfiltered: .9976 R^2 polynomial/filtered: .9976

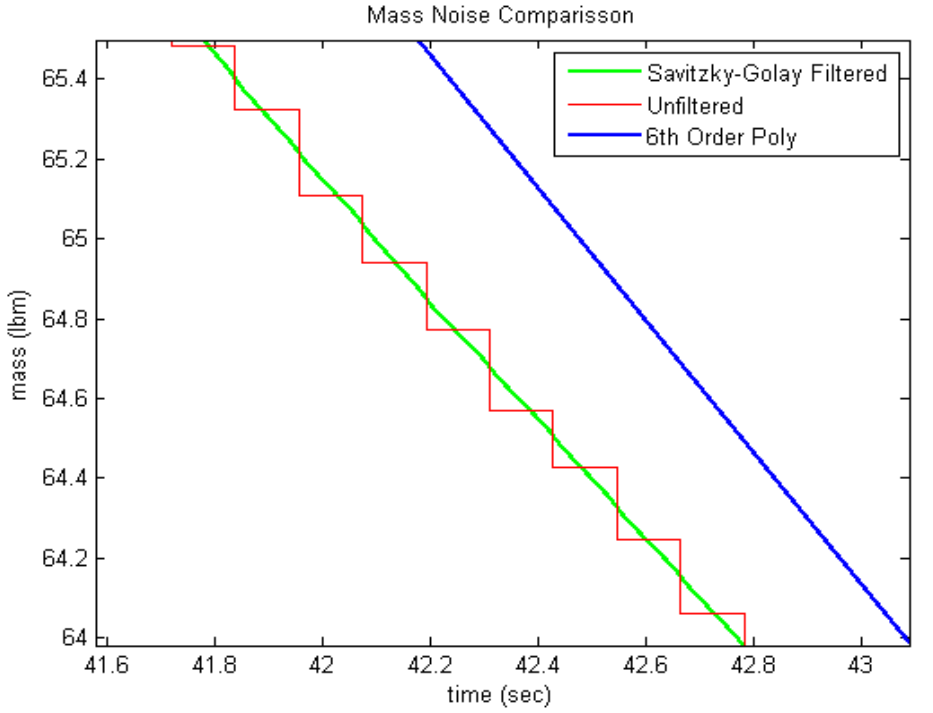
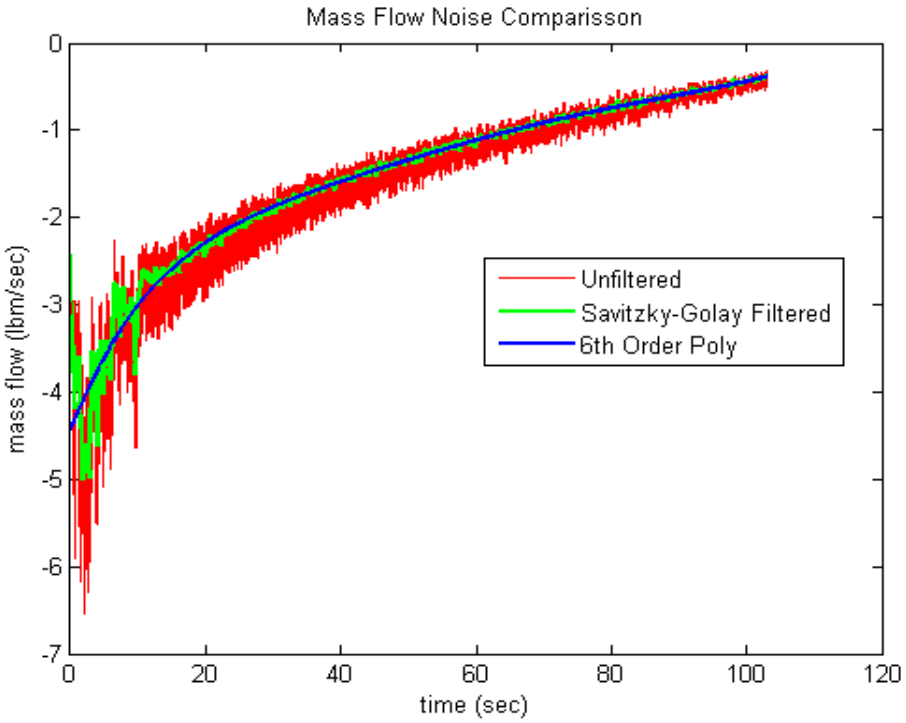


Figure 26: Mass Curve Noise (above) and Inset (below)



R^2 polynomial/unfiltered: .9160 R^2 polynomial/filtered: .9725

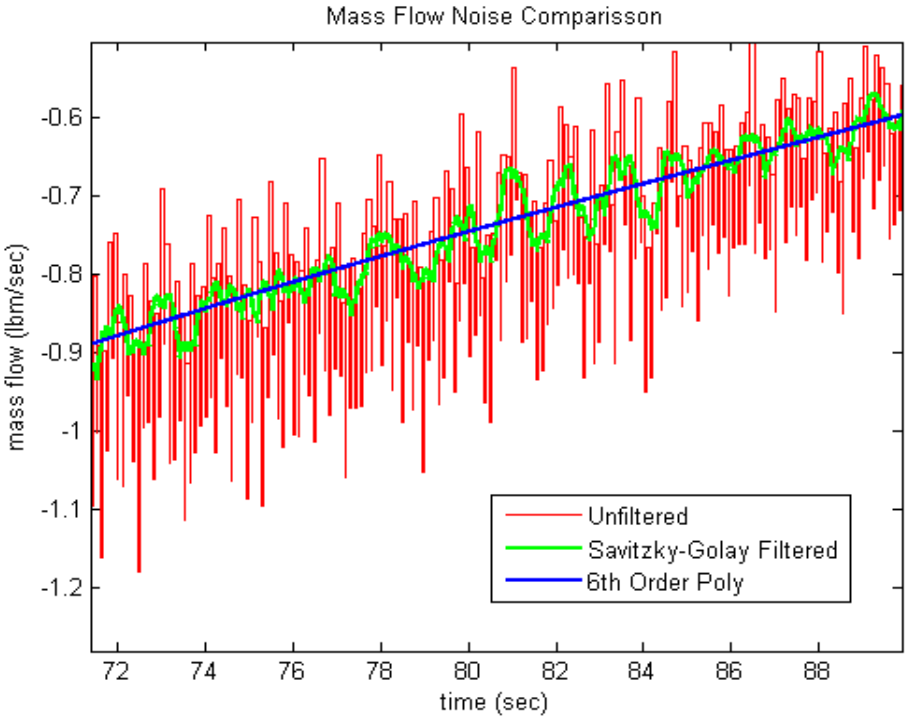


Figure 27: Mass Flow Curve Noise

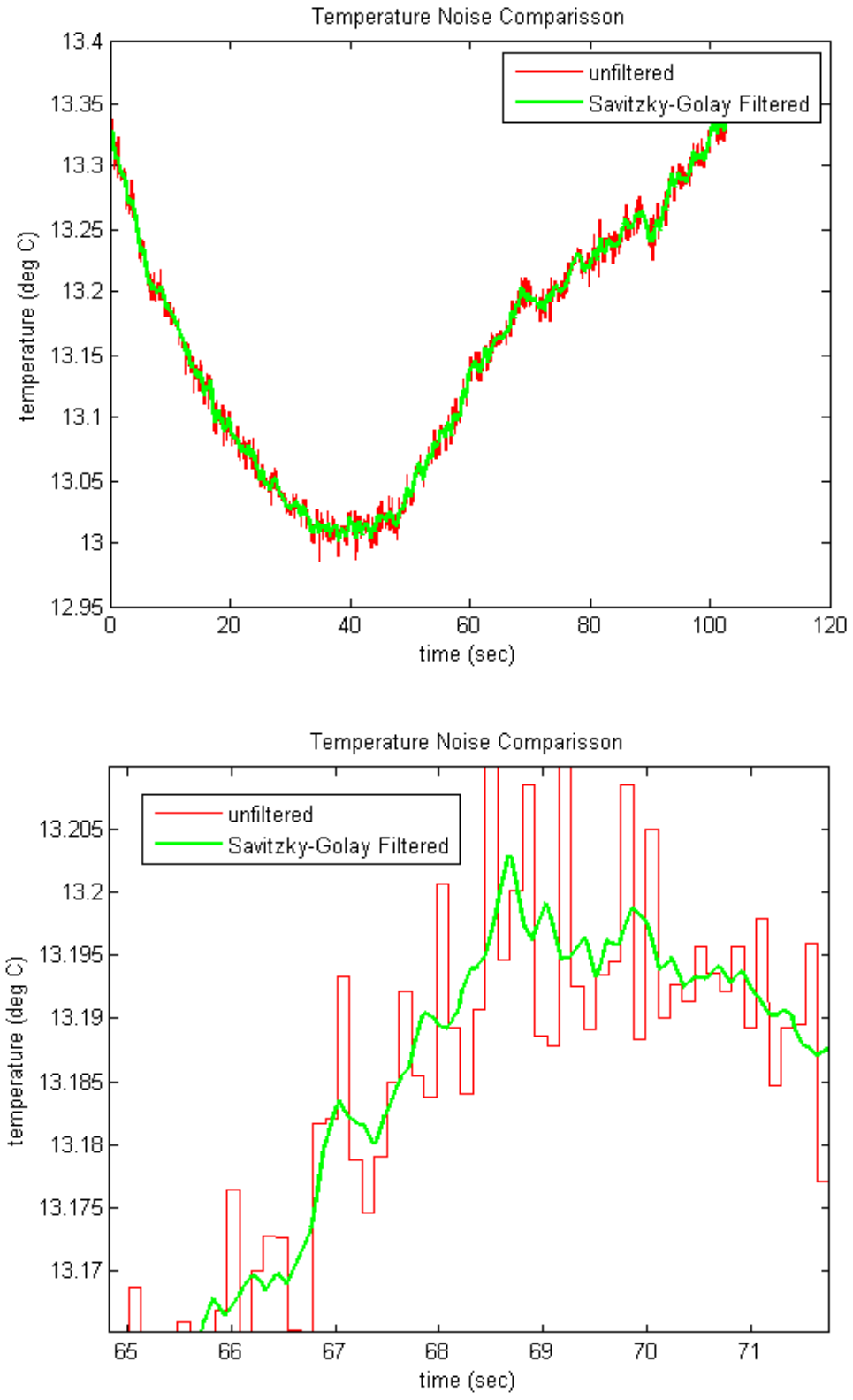


Figure 28: Temperature Noise

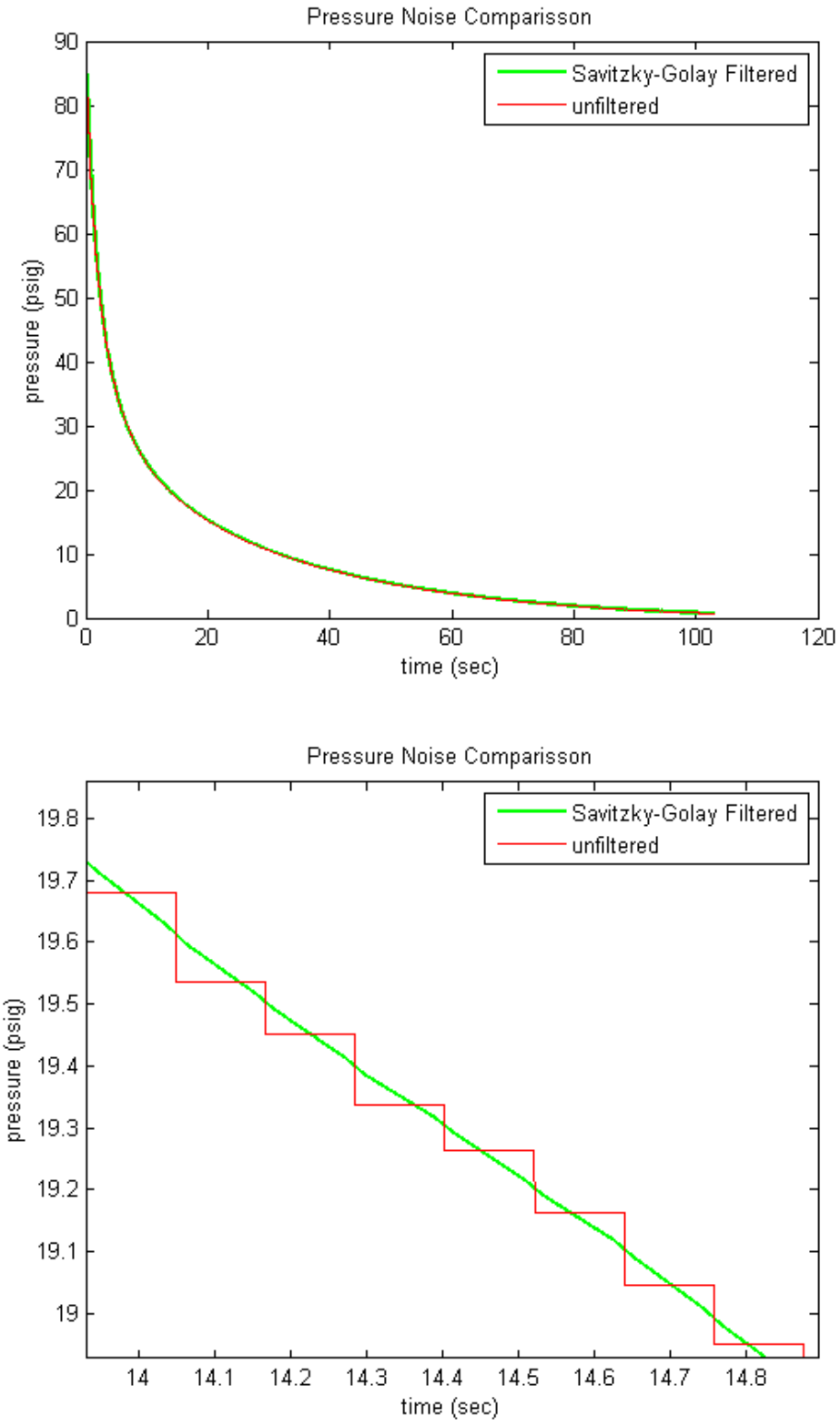


Figure 29: Pressure Noise

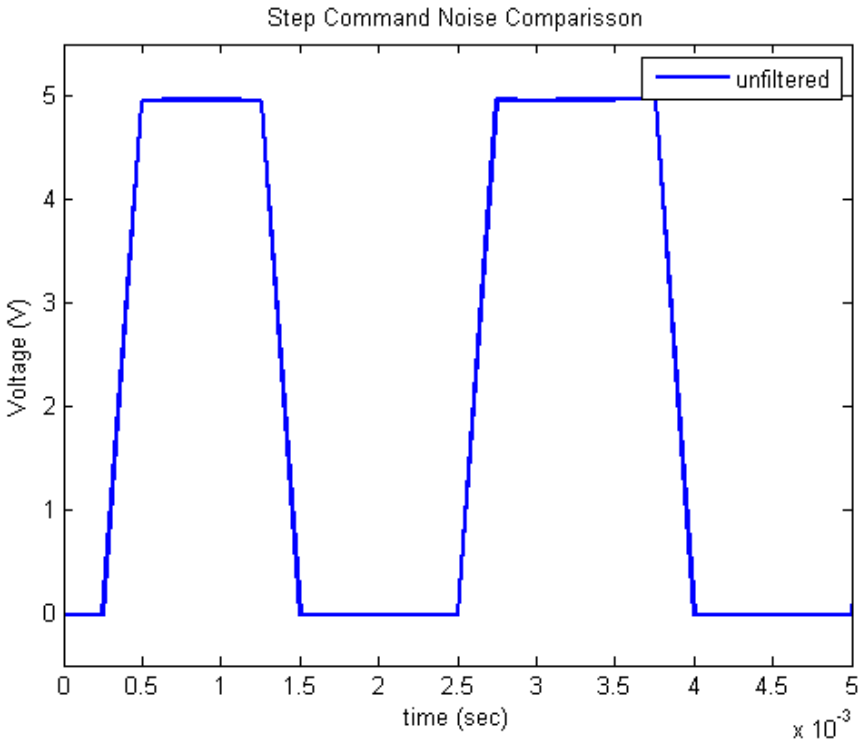


Figure 30: Step Command Noise

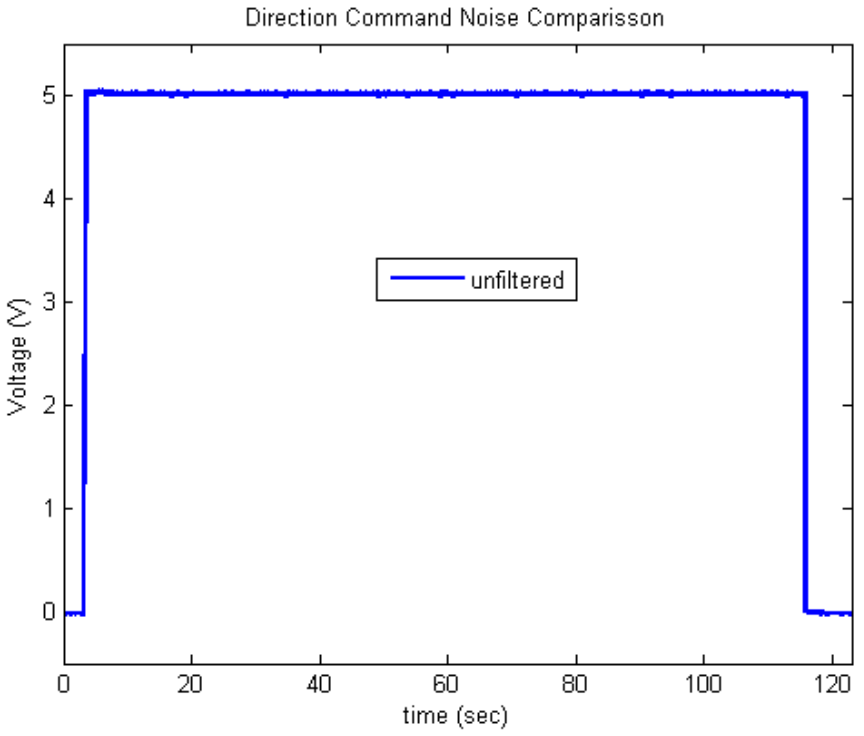


Figure 31: Direction Command Noise

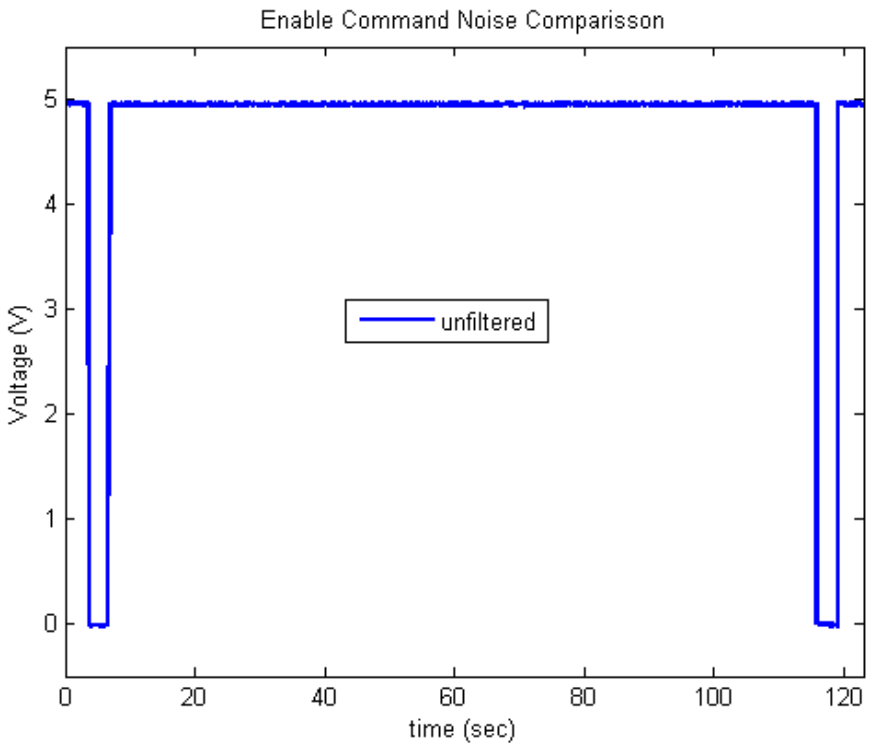


Figure 32: Enable Command Noise

6.2 Control System Analysis Post Processing

After a test of the control system has been conducted a very similar process in MATLAB is performed to extract the mass flow vs. time curve generated during the test run. This can be treated exactly like the mass flow vs. time curves during the characterization process, except that the valve position is not constant. Additionally, during the control system post processing the computer model is run such that a predicted mass flow curve for the given controller constants, initial pressures, and reference curve can be generated and then overlaid on the actual mass flow curve. This allows comparison between the model and reality. An Appendix C contains the MATLAB code, and remember that Appendix A contains the Simulink model used to model the system.

Once again, all of the same processes were used to measure the actual mass curve which gets compared to the predicted curve, except for the polynomial curve fitting. The mass flow curves with the controller proved too erratic to fit a curve reliably.

CHAPTER 7: RESULTS

This next section lays out the results of this thesis work divided between the results from the valve characterization and the results from the control system implementation.

7.1 Valve Characterization

The entire goal of the valve characterization portion of this thesis was to create a VF surface which could be used in the computer model of the oxidizer feed system. Figures 32-35 below display the finished VF surface. As a reminder, the value where VF is assigned to be 1 is for a valve position of 90deg and a pressure drop of 85psi. Once the VF plot was completed the VF multiplier was determined.

L=.147 lbm/sec

Additionally, to visualize the VF plot better Figures 36-44 show distinct slices of the VF surface. There are four plots of constant pressure drop (10, 20, 40, and 85psi) where the valve position changes, and another four plots where valve position is held constant (35, 55, 75, and 90deg) while pressure varies.

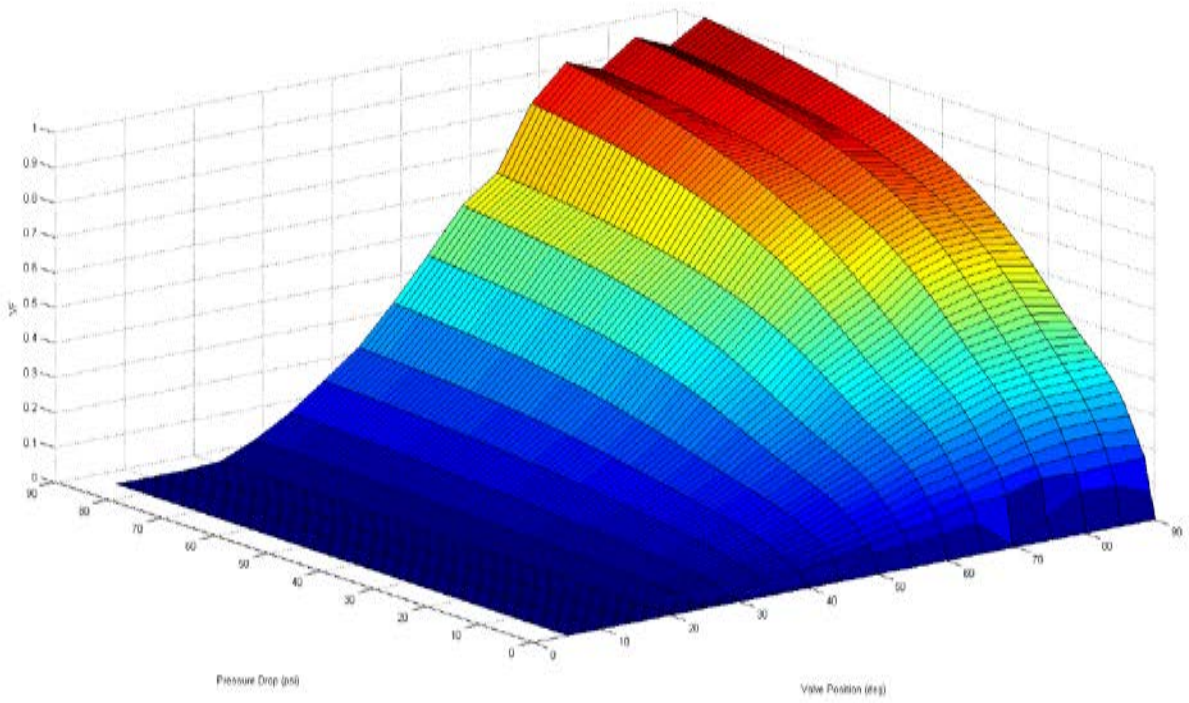


Figure 33: VF Plotted Against Valve Position and Pressure Drop

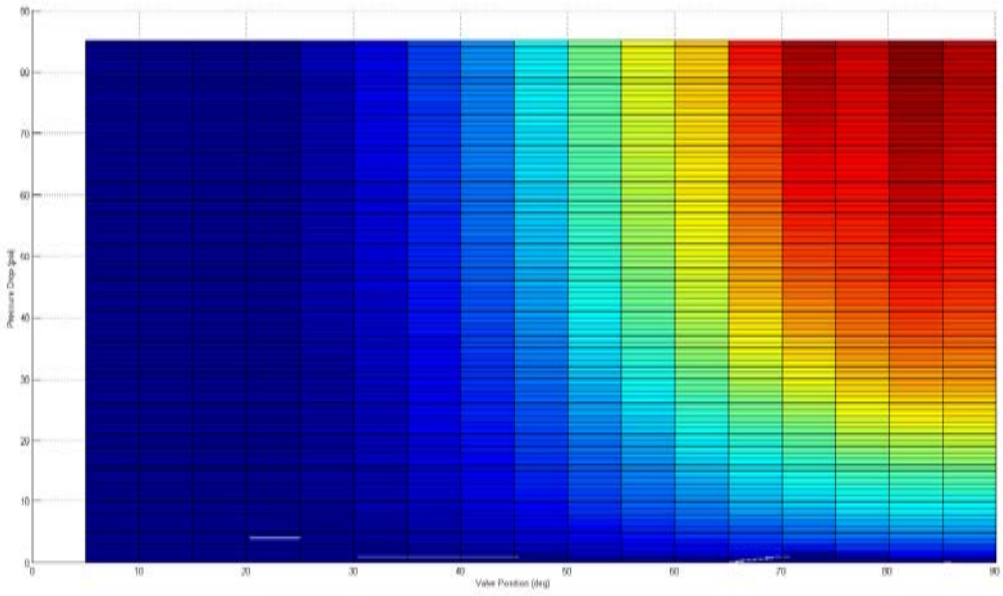


Figure 34: XY Plane of VF Surface

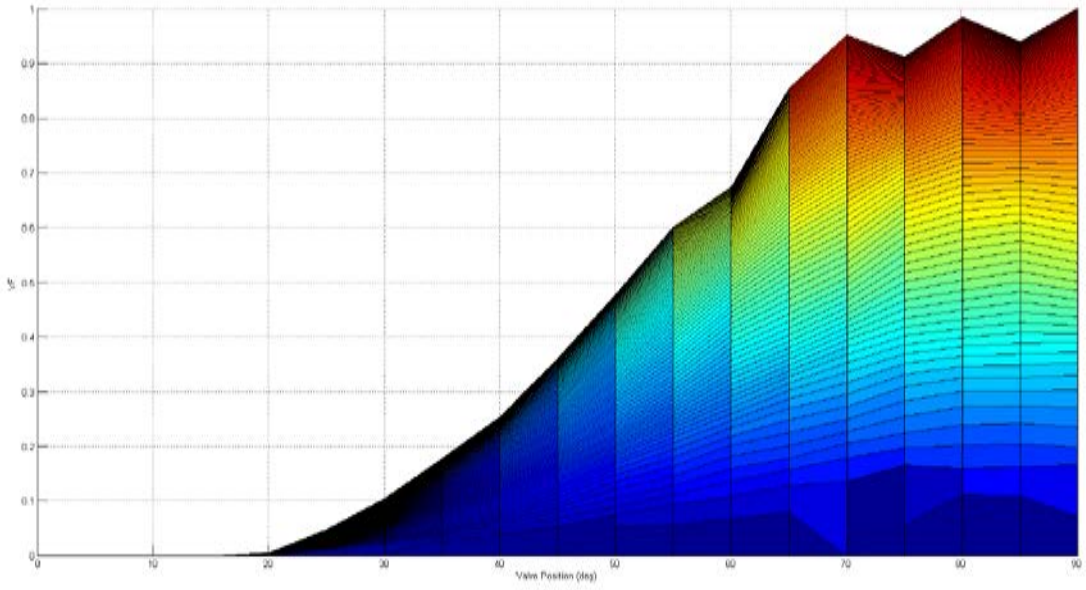


Figure 35: XZ Plane of VF Surface

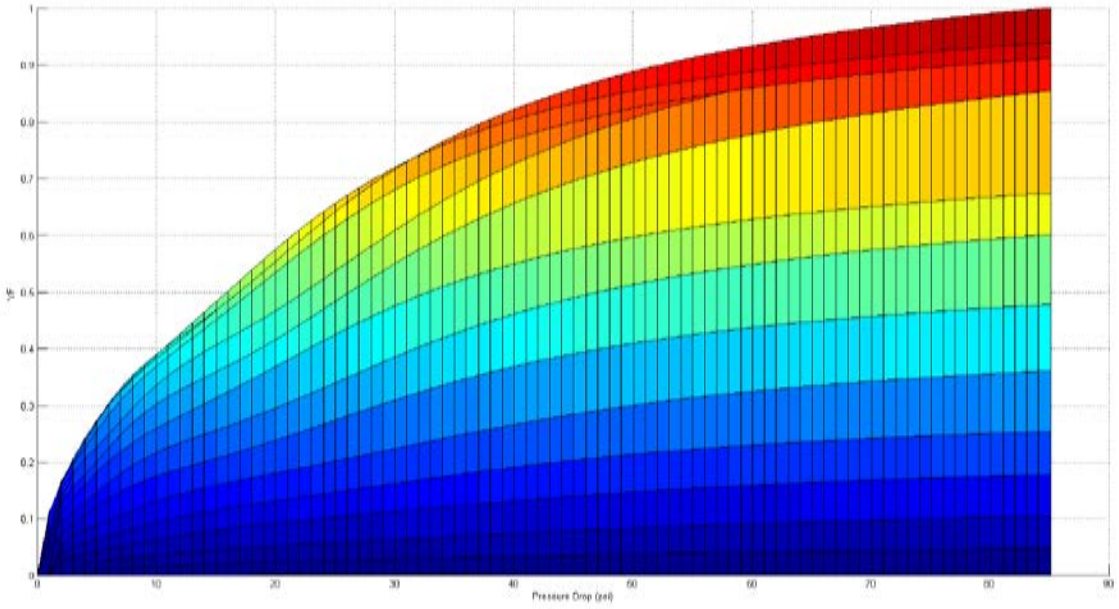


Figure 36: YZ Plane of VF Surface

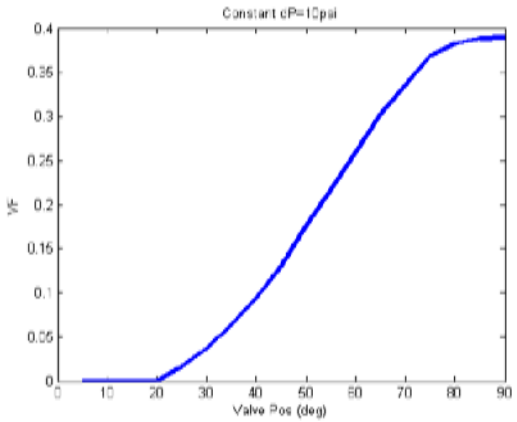


Figure 37: VF vs. Valve Position dP=10psi

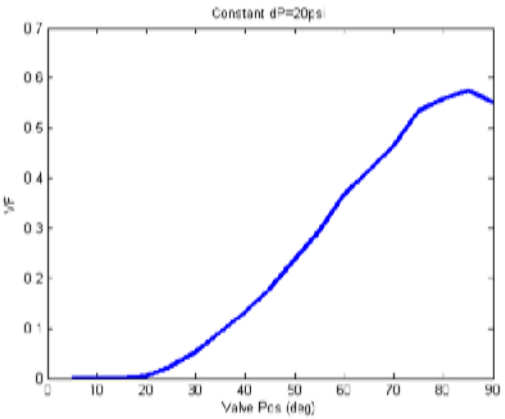


Figure 38: VF vs. Valve Position dP=20psi

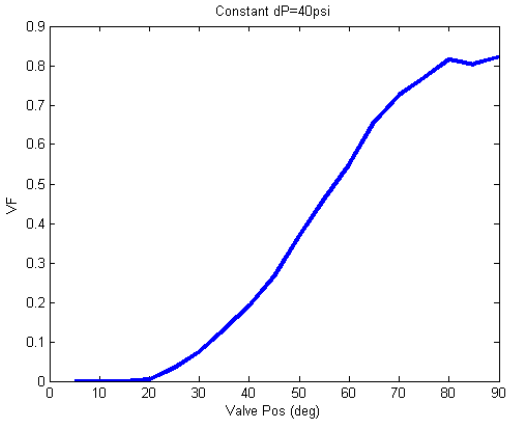


Figure 39: VF vs. Valve Position dP=40psi

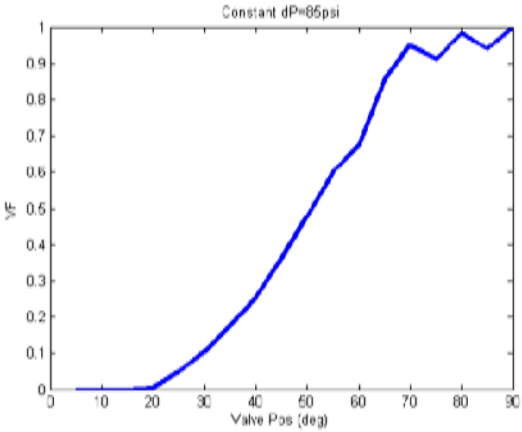


Figure 40: VF vs. Valve Position dP=85psi

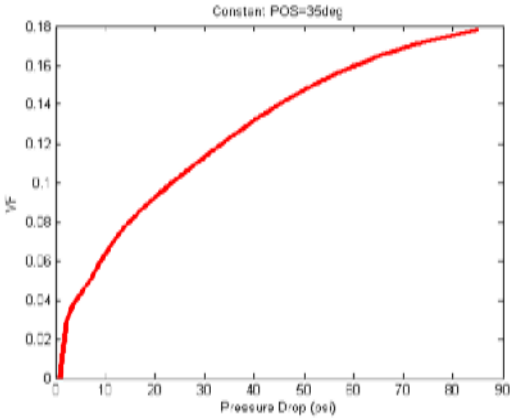


Figure 41: VF vs. Pressure Drop POS=35deg

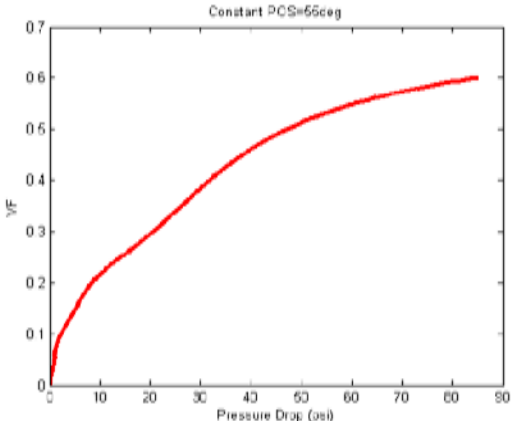


Figure 42: VF vs. Pressure Drop POS=55deg

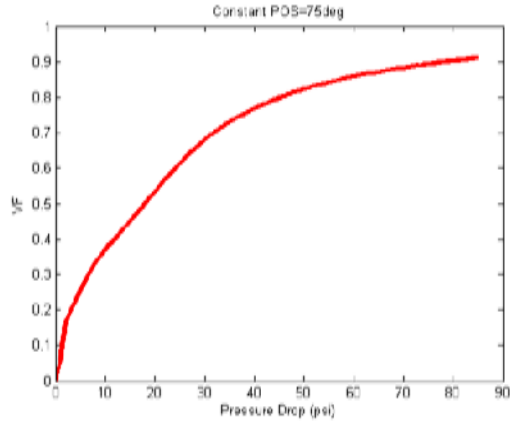


Figure 43: VF vs. Pressure Drop POS=75deg

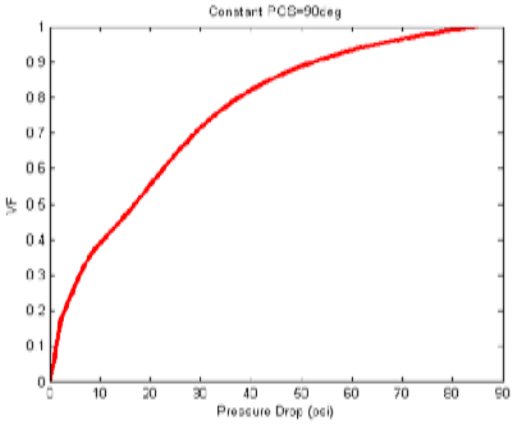


Figure 44: VF vs. Pressure Drop POS=90deg

7.2 Control System Results

The results from the control system testing were slightly less clean and conclusive, for a variety of reasons. They were very satisfactory for the initial investigation that this thesis provides. Only a P and PI controller were implemented, due to the impracticality of differentiating a noisy signal twice. There is still work to be done on the control system model, as it is the model works fairly well. It works best at higher values of KI and KP. The reason for the inaccuracies at the lower KI and KP values is not fully understood yet, however some theories are presented in the next section. The mass curves shown in the plots have all the filtering that was explained in the post processing section performed on them except for the polynomial curve fit. Their shapes proved too erratic to be fitted with a polynomial curve with any great deal of accuracy.

Figures 44-47 present some of the best examples, along with a couple which do not match up as well. Figures 48-59 show some plots of the predicted and measured tank pressure, which is an interesting side problem that ended up working out well. Next are two sets of plots that show 30 different simulations all side by side for comparison. Look for the explanation that follows each graph and explains how to read Figures 50 and 51.

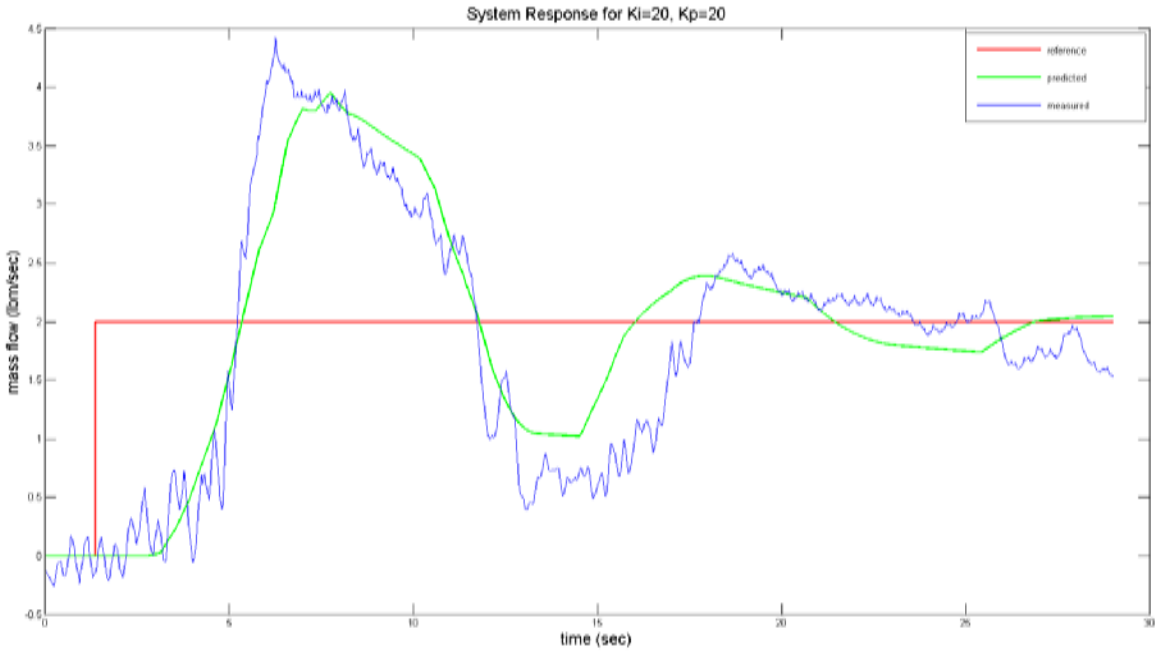


Figure 45: System Mass Flow Response for KI=20, KP=20

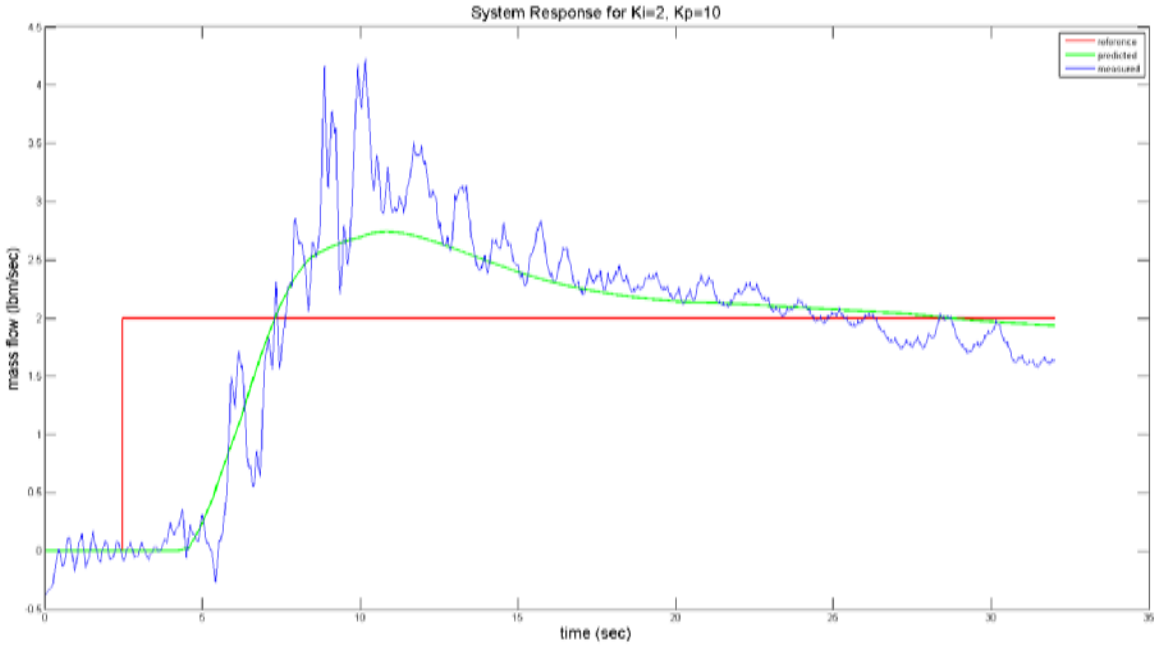


Figure 46: System Mass Flow Response for KI=5, KP=10

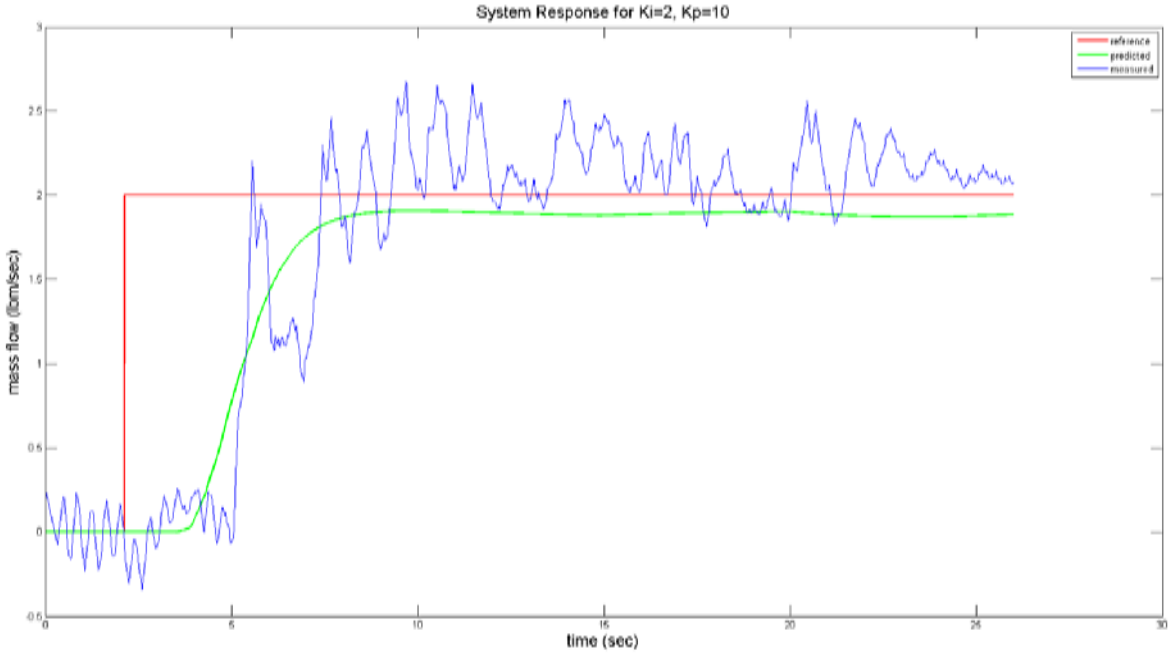


Figure 47: System Mass Flow Response for KI=2, KP=10

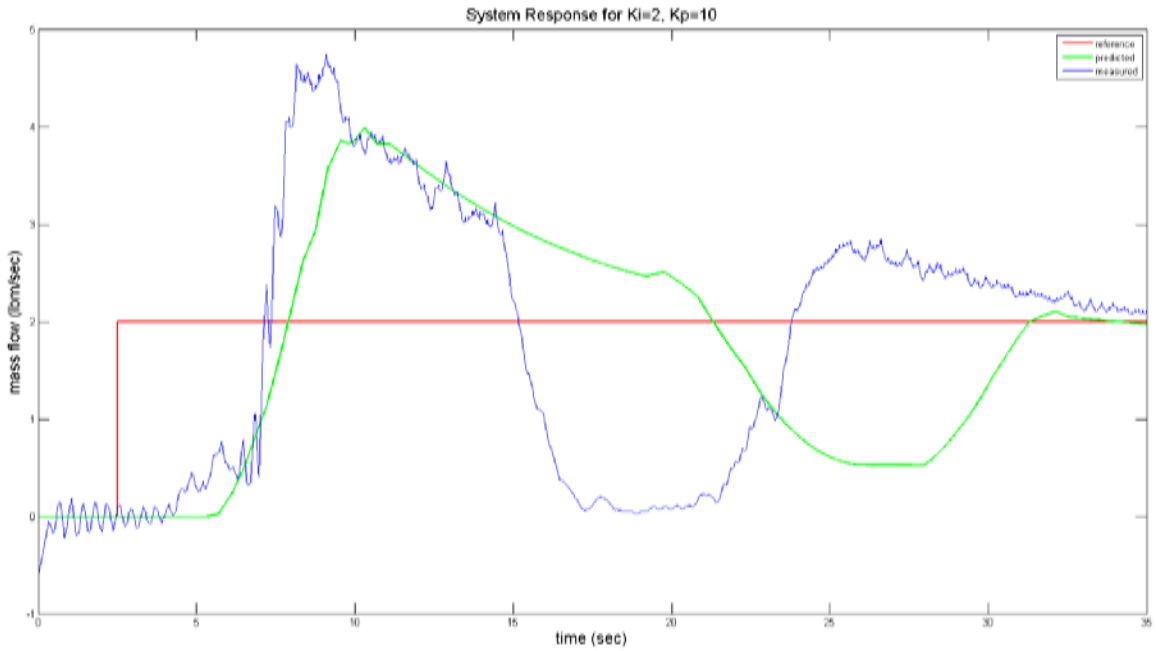


Figure 48: System Mass Flow Response for KI=5, KP=2

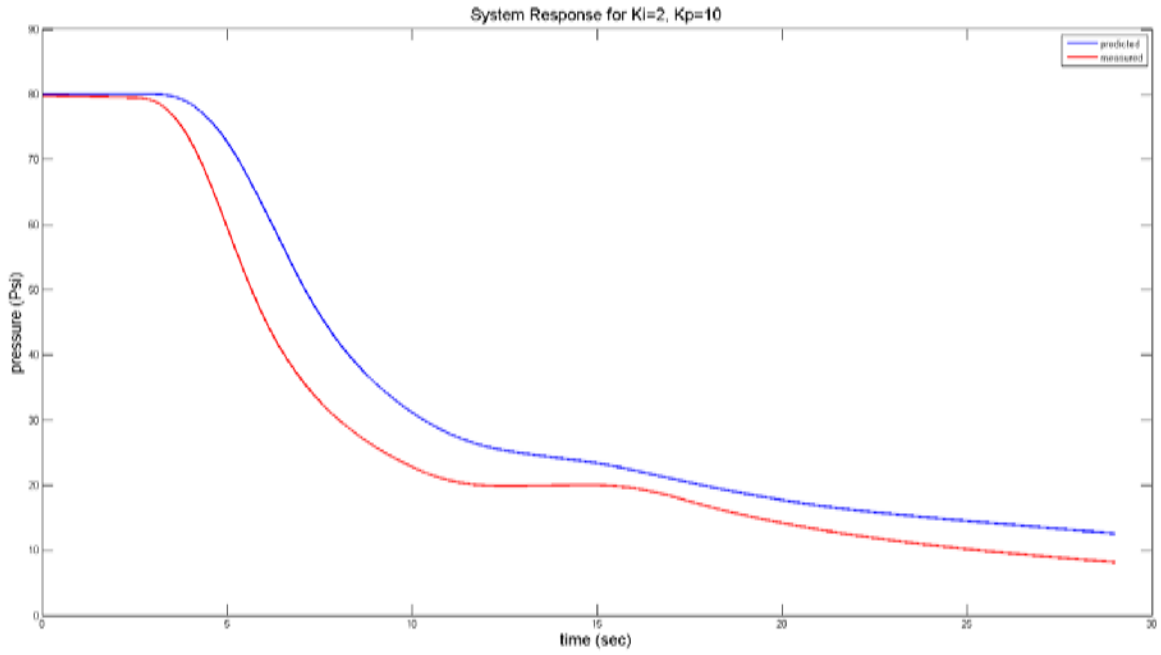


Figure 49: System Pressure Response for KI=5, KP=10

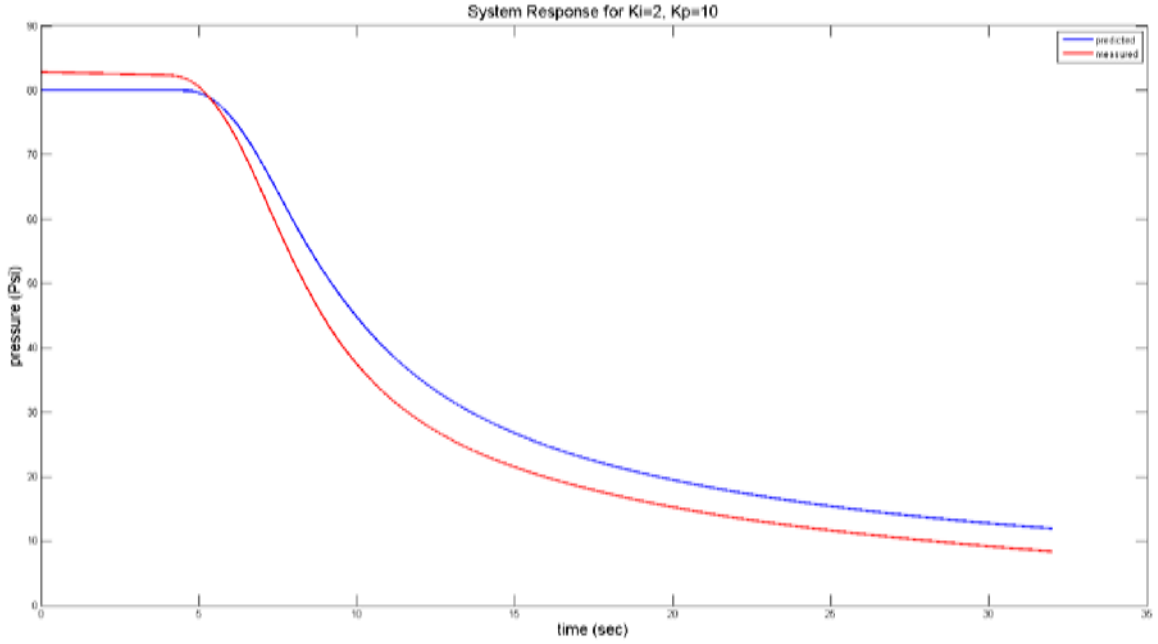


Figure 50: System Pressure Response for KI=2, KP=10

The following figures (50 and 51) require some explanation to read, since labeling all axes would clutter the figure. See the section immediately following the figures for a detailed description.

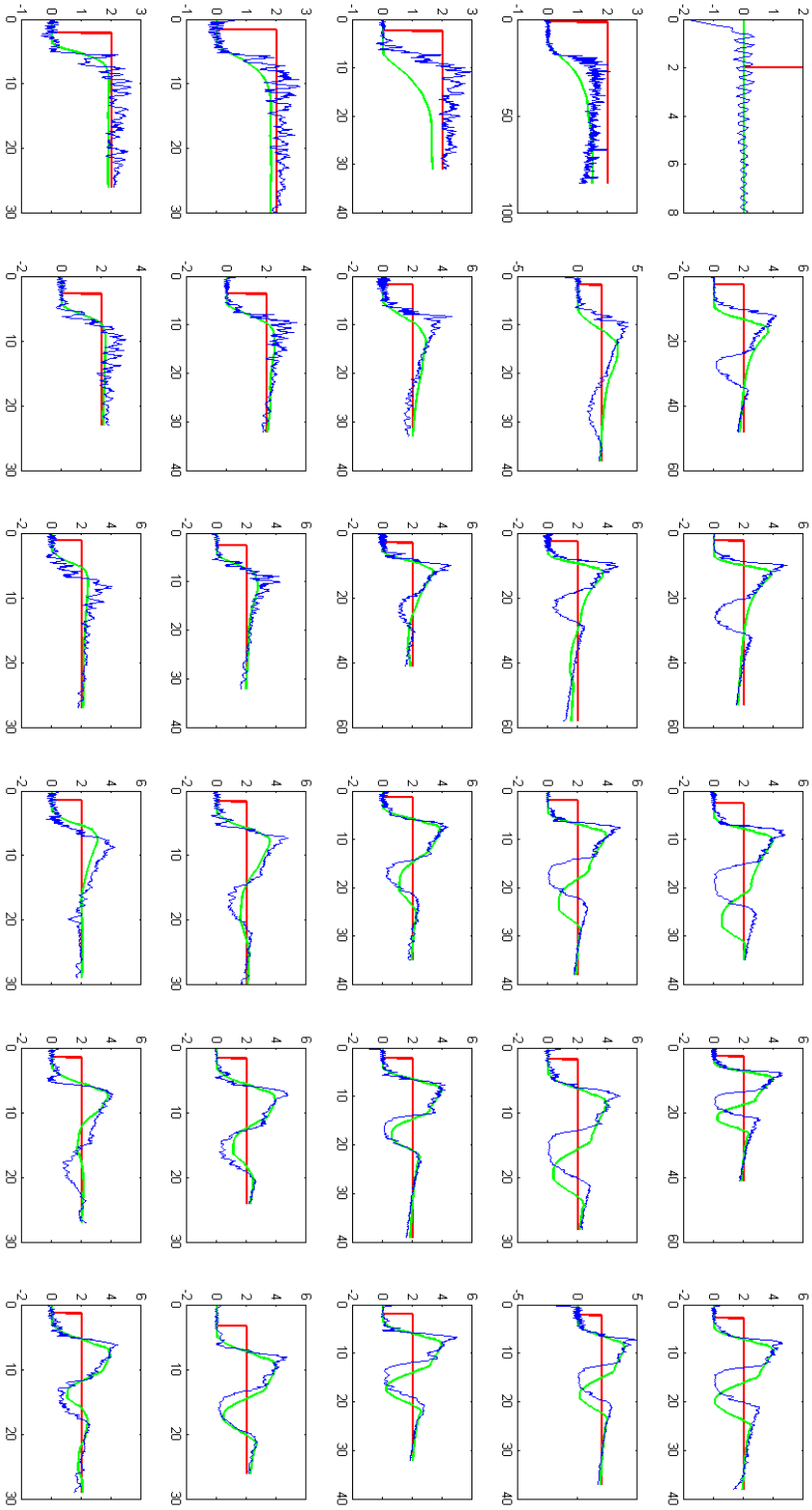


Figure 51: Variety of System Mass Flow Responses

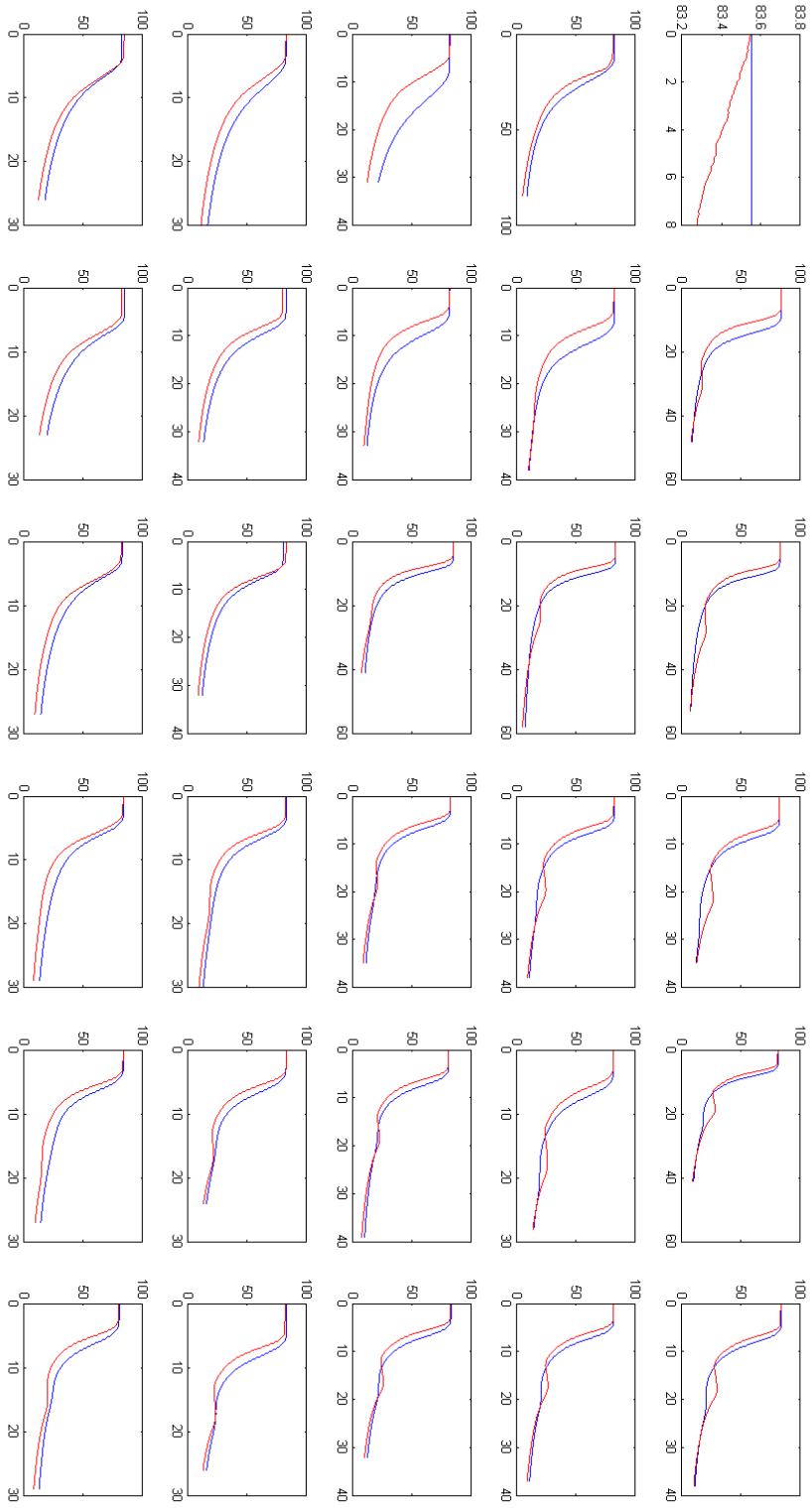


Figure 52: Variety of System Pressure Responses

Figures 50 and 51 require a little explanation. To pack the graphs as tightly together as possible there are no units or titles displayed. The horizontal axis is time (sec), just like Figures 44-49, the vertical axes are either mass flow in lbm/sec or pressure drop in psi (Figure 50 and 51 respectively). The top rows of graphs are for a KP of 0, second row for KP of 2, then 5, 10, and 20 respectively. The columns left to right are increasing KI, from 0, 1, 2, 5, 10, and 20. The red line signifies the reference signal, green is predicted response using the MATLAB simulation, and the blue is the measured system response. The pressure responses did not have a reference signal, therefore red is predicted and blue is measured.

CHAPTER 8: DISCUSSION OF RESULTS

As can be seen, not all of the results are in complete agreement with the model. However, there are patterns to the discrepancies and with more development time there is a very good chance to reconcile the model. There were several factors that contributed to the differences: Slop in the drivetrain, poor load cell amplification, 10-bit ADC, sloshing in the tank, a jittery direction signal, and errors in mass curve derivative to get mass flow.

Before each of the above is discussed it would be first useful to examine the pattern of inaccuracies seen in the data. If you read the description of Figures 50 and 51 such that you understand how KI and KP are organized over the rows and columns then it will be clear that the discrepancies are larger for the lower values of KI and KP. The model is the most inaccurate for very low values of KI (ie 1), and is also sensitive to the lower KP values, but not as much. Once KP is 5 or higher, and the KI is 2 or higher the model matches up very well.

Other inaccuracies are the result of bad resolution due to noise. This is most noticeable when the value of KI is lower; however the root cause is not the controller constants. The weak links in the system, such as the bad ADC, all contribute to a slightly noisy mass signal, which in turn make the mass flow curve even noisier. The controller is then using this noisy mass flow curve to calculate a response. With improved mass curve resolution the noise in the measured curve should be drastically improved. Keep in mind, however, that some of the noise displayed in the results is simply from the DAS, and does not precisely reflect actual mass flow. These difficulties have been discussed in the post-processing section of this report. The moving average implemented as a filter on the Arduino also helps with the noisy signal.

The Arduino only has 10-bit ADC, which converts the analog 0-5V signal to a digital signal between 0 and 1023. Since the amplifier on the load cell only reaches a value of .6V approximately the actual digital signal that reaches the Arduino is in the range of 0-130. This is not a great deal of resolution. In future tests it would be desirable to have at least a 12-bit ADC, possibly a 16-bit. Additionally, the amplifier itself on the load cell is lacking. If a true 0-5V signal could be attained then the resolution of the ADC would not be such a large issue.

Although this is accounted for in the simulation, the slop in the valve stem allows the motor to move momentarily without affecting the system. This causes additional oscillation of the motor which reduces its responsiveness. To solve this problem the valve must be changed. Most of the slop is the interface between the valve stem and the ball of the valve; which was not manufactured at Umaine. The parts that were custom fabricated for this project had very little play.

If the maximum possible valve angular velocity is changed in the computer model it is very easy to get the simulation and results to match up at any value of KI and KP. However, a certain angular velocity saturation does not fit all of the KI and KP values, therefore you can choose a value that makes the lower KI's and KP's match up or you can choose a value that makes the higher KI's and KP's match. This may indicate that the valve is not ever able to reach its maximum angular velocity with the lower KI and KP values, even though the simulation predicts that the error is high enough to trigger max velocity. One reason could be the noise in the signal, along with the slop in the drivetrain, is causing the valve to oscillate back and forth, which effectively reduces the max valve angular velocity. When KP and KI are higher, they are able to override this

tendency to reverse direction; allowing the valve velocity to be limited by the minimum step size command sent to the motor driver (which is the saturation level the simulation takes into account). Further investigation is required.

CHAPTER 9: CONCLUSION

The results of the work done match predictions to the extent that they prove feasibility of the initial idea, there are is some room for improvement. However, for a first pass the data and simulation were very satisfactory. Work should be done to reduce noise, and improve the quality of the mass flow vs. time curve such that it doesn't require such vigorous post processing. The physical build of the test apparatus held up very well, and the design proved to be robust and reliable, and the tank did not develop any leaks. The only problem with hardware was with the motor controller power supply, this burnt up despite being fuse protected. It was substituted with a computer power supply which worked wonderfully. There is no need to buy an expensive replacement power supply.

Next steps after improving the mass flow resolution would be to characterize a fluid similar to nitrous oxide, such as liquid CO2. Since carbon dioxide or nitrous oxide have a much lower vapor pressure they might flash when passing through the control valve which would reduce flow drastically as well as making the system nearly uncontrollable. This should be tested before this flow control system is used on a rocket motor, a different style valve might be necessary. However, the procedure established in this report should still be applicable. Once that has been done the only step needed to perform thrust control on a rocket motor would be to develop the conversion between chamber pressure and mass flow. This will introduce a whole other source of noise and error which will need to be addressed.

One area which was not explored at all in the scope of this thesis was the optimization of the control system. At this point there was not enough time to perform any optimization; however this would make an excellent future project. Now that the

simulation is working, a genetic algorithm would be the perfect optimization scheme, since this is such a complex system. A genetic algorithm would generate a large population of possible KI and KP combinations; run each one through the simulation to score their performance. Performance would most likely be the least square evaluation between the reference curve and the response. The best performing combinations or "individuals" would then be combined to generate new combinations or "children". There would be a chance of mutation in the process of creating the children. These new combinations would then be run through the simulation to be given their own score, the process repeats until the scores level off.

Eventually this technology should be applied on an actual rocket engine. With the development of a throttled hybrid motor it would make the creation of a reusable sounding rocket cheaper and more adaptable, especially for launching fragile payloads such as Cube Satellites. The Cube Satellite field has generated much interest in the smaller vehicles. The team Ursa DeltaP sounding rocket which is being developed in parallel to this thesis is an excellent example. The rocket was designed such that a hybrid motor could be easily retrofitted as a booster. A maiden launch of the Delta P is scheduled for July 27, 2012. For this first launch two solid propellant motors will be used. The future of small LEO and suborbital launch vehicles is generally looking very bright. Companies and individuals are realizing that the costs of Boeing and Lockheed launch vehicles (Delta II's, Atlas, Delta IV's, etc) are much too high and often overkill. With cost saving technologies expanding in the small aerospace field more individuals and small companies can afford to do research at high altitude and supersonic speeds.

APPENDIX A: SIMULINK MODEL SCREENSHOTS

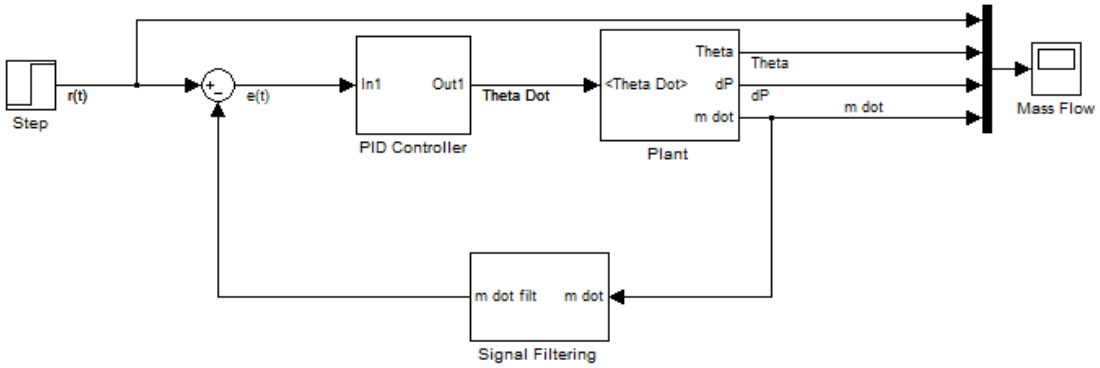


Figure A1: Overall Block Diagram

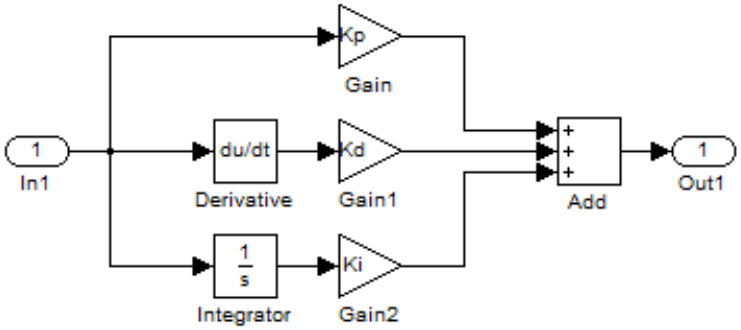


Figure A2: PID Controller Block

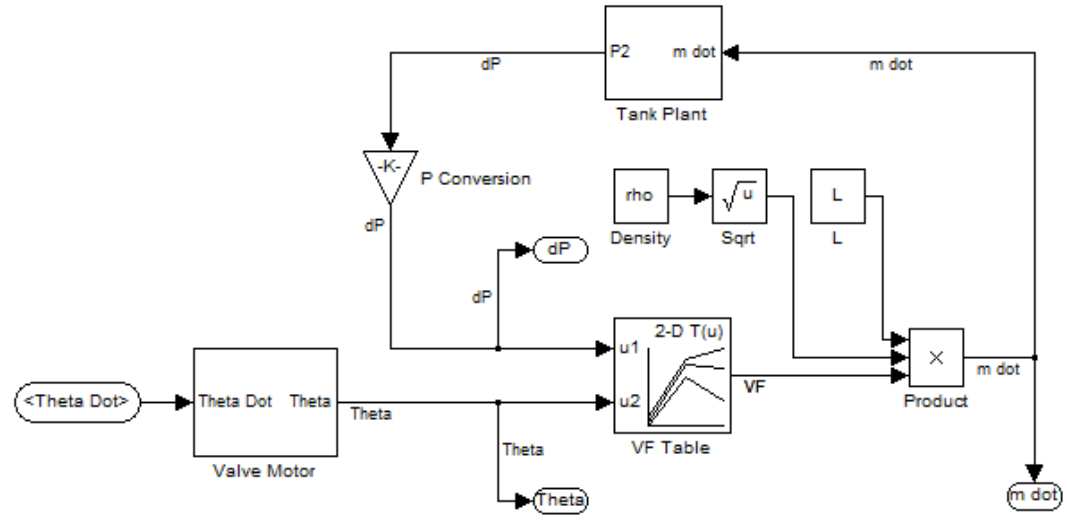


Figure A3: Plant Block

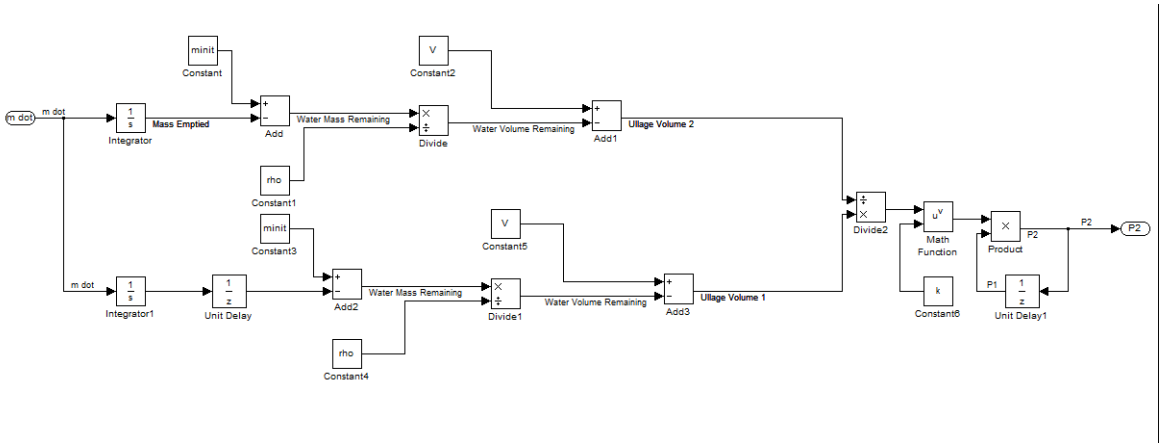


Figure A4: Tank Plant Block

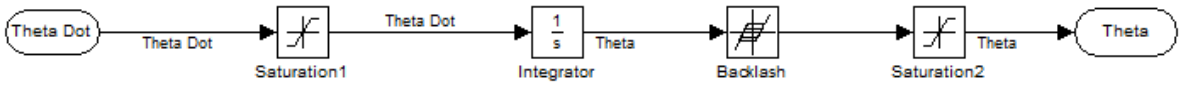


Figure A5: Valve Motor Block

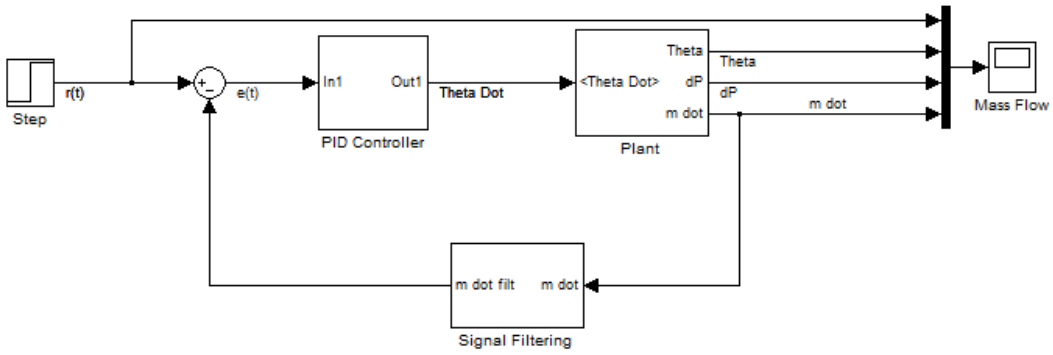


Figure A6: Signal Filtering Block

APPENDIX B: VF PLOT POST PROCESSING MATLAB CODE

```
clear all
close all
clc

tstart=[0,0,0,0,22,22,16,14,10,9,11,10,9,9,8.5,7,7,6]; %start keeping data (sec)
tend=[18,20,20,4500,1250,630,510,320,270,240,200,160,140,113,113,130,110,110]; %finish
keeping data (sec)
theta=[5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90];
pmax=85; %max pressure for valid data (psi)
dp=1; %pressure steps in VF table (psi)
n=1;

for j=1:n

    data=[];
    DATA=[];
    DATAcrop=[];

    %Read in Data

    if j==1
        data=dlmread('Output DATA_5deg_150psi.lvm');
    end
    if j==2
        data=dlmread('Output DATA_10deg_150psi.lvm');
    end
    if j==3
        data=dlmread('Output DATA_15deg_150psi.lvm');
    end
    if j==4
        data=dlmread('Output DATA_20deg_150psi.lvm');
    end
    if j==5
        data=dlmread('Output DATA_25deg_150psi.lvm');
    end
    if j==6
        data=dlmread('Output DATA_30deg_150psi.lvm');
    end
    if j==7
        data=dlmread('Output DATA_35deg_150psi.lvm');
    end
    if j==8
        data=dlmread('Output DATA_40deg_150psi.lvm');
    end
    if j==9
        data=dlmread('Output DATA_45deg_150psi.lvm');
    end
    if j==10
        data=dlmread('Output DATA_50deg_150psi.lvm');
    end
    if j==11
```

```
    data=dlmread('Output DATA_55deg_150psi.lvm');
end
if j==12
    data=dlmread('Output DATA_60deg_150psi.lvm');
end
if j==13
    data=dlmread('Output DATA_65deg_150psi.lvm');
end
if j==14
    data=dlmread('Output DATA_70deg_150psi.lvm');
end
if j==15
    data=dlmread('Output DATA_75deg_150psi.lvm');
end
if j==16
    data=dlmread('Output DATA_80deg_150psi.lvm');
end
if j==17
    data=dlmread('Output DATA_85deg_150psi.lvm');
end
if j==18
    data=dlmread('Output DATA_90deg_170psi.lvm');
end

%get time step

dt=data(2,1)-data(1,1);

%remove steps in the data for temp, pressure, and mass

DATA(:,1)=data(:,1);
DATA(:,2)=sgolayfilt(data(:,2),1,2001);
DATA(:,3)=sgolayfilt(data(:,3),1,2001);
DATA(:,4)=sgolayfilt(data(:,4),1,2001);
DATA(:,5)=data(:,5);
DATA(:,6)=data(:,6);
DATA(:,7)=data(:,7);

%differentiate mass curve and filter

DATA(:,8)=[(diff(DATA(:,3))/dt);0];
DATA(:,9)=sgolayfilt(DATA(:,8),1,2001);

%Generate angle data

count=0;
for i=1:(length(DATA(:,1))-1)
    if (DATA(i,6)<=2.5)
        dir=-1;
    end
    if (DATA(i,6)>2.5)
```

```
            dir=1;
        end
        if ((abs((DATA(i,5)-DATA((i+1),5)))>2.5)&&(DATA(i,7)<2.5))
            count=count+dir;
        end
        DATA(i,10)=(count/2)*(1.8/30);
    end

    %Resize DATA matrix

    DATAcrop(:,1)=[0:dt:(tend(j)-tstart(j))];
    DATAcrop(:,2)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),2);
    DATAcrop(:,3)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),3);
    DATAcrop(:,4)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),4);
    DATAcrop(:,5)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),5);
    DATAcrop(:,6)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),6);
    DATAcrop(:,7)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),7);
    DATAcrop(:,8)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),8);
    DATAcrop(:,9)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),9);
    DATAcrop(:,10)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),10);

    %Polynomial fit for mass and mass flow

    P=polyfit(DATAcrop(:,1),DATAcrop(:,3),2);
    for i=1:length(DATAcrop(:,1))
        DATAcrop(i,11)=P(1)*(DATAcrop(i))^2+P(2)*(DATAcrop(i))+P(3);
    end
    P=polyfit(DATAcrop(:,1),DATAcrop(:,9),6);
    for i=1:length(DATAcrop(:,1))
        DATAcrop(i,12)=P(1)*(DATAcrop(i))^6+P(2)*(DATAcrop(i))^5+P(3)*(DATAcrop(i))^4+P↙
(4)*(DATAcrop(i))^3+P(5)*(DATAcrop(i))^2+P(6)*(DATAcrop(i))+P(7);
    end

    %Generate new table for mass flow vs. pressure

    DATAcrop(:,13)=(round(DATAcrop(:,4)*(1/(dp/10)))/(1/(dp/10)));
    for i=1:((pmax/dp)+1)
        MvP(i,1,j)=(i-1)*dp;
        MvP(i,2,j)=(-1)*mean(DATAcrop((find(DATAcrop(:,13)==MvP(i,1,j))),12));
        if (isnan(MvP(i,2,j))==1)
            MvP(i,2,j)=0;
        end
    end
    j
end

for i=1:n
    plot(MvP(:,1,i),MvP(:,2,i))
    hold on
end
```

```
for i=1:n
    VF(:,i)=MvP(:,2,i);
end

VF(:,:)=VF(:,:)/VF(86,18);

surf(theta,MvP(:,1,1),VF)
xlabel('Valve Position (deg)')
ylabel('Pressure Drop (psi)')
zlabel('VF')
```

APPENDIX C: CONTROLLER ANALYSIS POST PROCESSING MATLAB CODE

```
clear all
close all
clc

tstart=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]; %start keeping data↙
(sec)
tend=↙
[15,55,60,42,48,45,92,45,65,45,35,44,38,40,48,42,46,39,37,39,39,37,31,33,33,30,34,36,34,3↙
6]; %finish keeping data (sec)
tend=tend-7;
massdotstart=[2,2.6,2.4,2.5,2.64,2.86,1.47,1.75,2.6,1.91,1.84,2.2,2.43,1.74,...
    2.93,1.28,2.21,1.85,1.58,3.68,2.48,1.67,1.62,3.26,2.13,2.76,1.15,1.43,1.52,1.38];
KI=[0 1 2 5 10 20 0 1 2 5 10 20 0 1 2 5 10 20 0 1 2 5 10 20 0 1 2 5 10 20];
KP=[0 0 0 0 0 0 2 2 2 2 2 2 5 5 5 5 5 5 10 10 10 10 10 10 20 20 20 20 20 20];
Pressureinit=[83.55 83.5 82.5 82.45 80.9 83.5 82 82 83 81.8 81 81.5 ...
    81.5 81.5 84.5 82 80.5 82.5 82.5 82.5 80 82.8 82.5 83 82 84.5 82.5 83.5 83.5 80];
massdot=2;
m=30; %first plot to display
n=30; %last plot to display

%%%SIMULINK SIMULATION PARAMETERS%%%

DT=.0001; %Simulation Duration (sec)

%%%SIMULINK PHYSICAL SYSTEM PARAMETERS%%%

Dthetamax=13; %Max valve position rate of change (deg/s)
Thetamax=90; %Max valve position, 90=open (deg)
Thetamin=0; %Min valve position, 0=closed (deg)
ratio=30; %Gearbox ratio
slop=180; %Drivetrain slop angle at motor shaft (deg)
L=.0672698678; %VF Multiplier
V=.0761; %Tank Volume (m^3)
R=296.8; %Ullage Ideal Gas Constant (kJ/kg-K)
k=1.4; %Ratio of specific heats for nitrogen (unitless)
r=.857; %Correction of k since system is not 100% adiabatic (where k=1 is isothermal)
g=9.807; %Gravitational Acceleration (m/s^2)
rho=1000; %Density of Water (kg/m^3)
load 'MvP.mat'
load 'VF.mat'
load 'theta.mat'

%%%INITIAL CONDITIONS%%%
Vinit=.91079*V; %Initial Water Volume (m^3)
Tinit=298; %Ullage Initial Temperature (K)

%%%CALCULATED INITIAL CONDITIONS%%
minit=Vinit*rho; %Initial Water Mass (kg)
Vullinit=V-Vinit; %Initial Ullage Volume (m^3)

for j=m:n
```

```
MASSDOT=massdot;
MASSDOTSTART=massdotstart(j);

data=[];
DATA=[];
DATAcrop=[];

%Read in Data

if j==1
    data=dlmread('Controller_KI0_KP0_P80_M2_MOD6.lvm');
end
if j==2
    data=dlmread('Controller_KI1_KP0_P80_M2_MOD6.lvm');
end
if j==3
    data=dlmread('Controller_KI2_KP0_P80_M2_MOD6.lvm');
end
if j==4
    data=dlmread('Controller_KI5_KP0_P80_M2_MOD6.lvm');
end
if j==5
    data=dlmread('Controller_KI10_KP0_P80_M2_MOD6.lvm');
end
if j==6
    data=dlmread('Controller_KI20_KP0_P80_M2_MOD6.lvm');
end
if j==7
    data=dlmread('Controller_KI0_KP2_P80_M2_MOD6.lvm');
end
if j==8
    data=dlmread('Controller_KI1_KP2_P80_M2_MOD6.lvm');
end
if j==9
    data=dlmread('Controller_KI2_KP2_P80_M2_MOD6.lvm');
end
if j==10
    data=dlmread('Controller_KI5_KP2_P80_M2_MOD6.lvm');
end
if j==11
    data=dlmread('Controller_KI10_KP2_P80_M2_MOD6.lvm');
end
if j==12
    data=dlmread('Controller_KI20_KP2_P80_M2_MOD6.lvm');
end
if j==13
    data=dlmread('Controller_KI0_KP5_P80_M2_MOD6.lvm');
end
if j==14
    data=dlmread('Controller_KI1_KP5_P80_M2_MOD6.lvm');
```

```
end
if j==15
    data=dlmread('Controller_KI2_KP5_P80_M2_MOD6.lvm');
end
if j==16
    data=dlmread('Controller_KI5_KP5_P80_M2_MOD6.lvm');
end
if j==17
    data=dlmread('Controller_KI10_KP5_P80_M2_MOD6.lvm');
end
if j==18
    data=dlmread('Controller_KI20_KP5_P80_M2_MOD6.lvm');
end
if j==19
    data=dlmread('Controller_KI0_KP10_P80_M2_MOD6.lvm');
end
if j==20
    data=dlmread('Controller_KI1_KP10_P80_M2_MOD6.lvm');
end
if j==21
    data=dlmread('Controller_KI2_KP10_P80_M2_MOD6.lvm');
end
if j==22
    data=dlmread('Controller_KI5_KP10_P80_M2_MOD6.lvm');
end
if j==23
    data=dlmread('Controller_KI10_KP10_P80_M2_MOD6.lvm');
end
if j==24
    data=dlmread('Controller_KI20_KP10_P80_M2_MOD6.lvm');
end
if j==25
    data=dlmread('Controller_KI0_KP20_P80_M2_MOD6.lvm');
end
if j==26
    data=dlmread('Controller_KI1_KP20_P80_M2_MOD6.lvm');
end
if j==27
    data=dlmread('Controller_KI2_KP20_P80_M2_MOD6.lvm');
end
if j==28
    data=dlmread('Controller_KI5_KP20_P80_M2_MOD6.lvm');
end
if j==29
    data=dlmread('Controller_KI10_KP20_P80_M2_MOD6.lvm');
end
if j==30
    data=dlmread('Controller_KI20_KP20_P80_M2_MOD6.lvm');
end

%get time step
```

```
dt=data(2,1)-data(1,1);

%remove steps in the data for temp, pressure, and mass

DATA(:,1)=data(:,1);
DATA(:,2)=sgolayfilt(data(:,2),1,2001);
DATA(:,3)=sgolayfilt(data(:,3),1,2001);
DATA(:,4)=sgolayfilt(data(:,4),1,2001);
DATA(:,5)=data(:,5);
DATA(:,6)=data(:,6);
DATA(:,7)=data(:,7);

%differentiate mass curve and filter

DATA(:,8)=[(diff(DATA(:,3))/dt);0];
DATA(:,9)=sgolayfilt(DATA(:,8),1,2001);

%Generate angle data

count=0;
for i=1:(length(DATA(:,1))-1)
    if (DATA(i,6)<=2.5)
        dir=-1;
    end
    if (DATA(i,6)>2.5)
        dir=1;
    end
    if ((abs((DATA(i,5)-DATA((i+1),5)))>2.5)&&(DATA(i,7)<2.5))
        count=count+dir;
    end
    DATA(i,10)=(count/2)*(1.8/30);
end

%Resize DATA matrix

DATAcrop(:,1)=[0:dt:(tend(j)-tstart(j))];
DATAcrop(:,2)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),2);
DATAcrop(:,3)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),3);
DATAcrop(:,4)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),4);
DATAcrop(:,5)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),5);
DATAcrop(:,6)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),6);
DATAcrop(:,7)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),7);
DATAcrop(:,8)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),8);
DATAcrop(:,9)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),9);
DATAcrop(:,10)=DATA(((tstart(j)/dt)+1):((tend(j)/dt)+1),10);

%%%SIMULINK SIMULATION PARAMETERS%%%

Dur=tend(j); %Simulation Duration (sec)
```

```
    %%%INITIAL CONDITIONS%%%

    Pinit=Pressureinit(j)*6894.7573; %Ullage Initial Pressure (Pa)

    %%%CONTROLLER PARAMETERS%%%

    Kp=KP(j);
    Kd=0;
    Ki=KI(j);

    %%%CALCULATED INITIAL CONDITIONS%%

    mull=(Pinit*Vullinit)/(R*Tinit); %Initial Ullage Mass (kg)

    %%%RUN SIMULATION%%%

    sim('SystemSimulation.mdl')

    %%%PLOTTING%%%

    figure(1)
    subplot(5,6,j)
    plot(ScopeData(:,1),2.204622*ScopeData(:,2),'red','LineWidth',2)
    hold on
    plot(ScopeData(:,1),2.204622*ScopeData(:,3),'green','LineWidth',2)
    hold on
    plot(DATAcrop(:,1),(-1)*DATAcrop(:,9),'blue')
    xlabel('time (sec)','fontsize',16)
    ylabel('mass flow (lbm/sec)','fontsize',16)
    title('System Response for Ki=2, Kp=10','fontsize',16)
    legend('reference','predicted','measured','fontsize',14)
    hold on

    figure(2)
    subplot(5,6,j)
    plot(ScopeData(:,1),ScopeData(:,5),'LineWidth',2)
    hold on
    plot(DATAcrop(:,1),DATAcrop(:,4),'red','LineWidth',2)
    xlabel('time (sec)','fontsize',16)
    ylabel('pressure (Psi)','fontsize',16)
    title('System Response for Ki=2, Kp=10','fontsize',16)
    legend('predicted','measured','fontsize',14)
    hold on
    j
end
```

APPENDIX D: MANUAL ARDUINO CODE

This program is used to allow manual control of opening or closing the valve, and is the most basic of the three programs. Understanding the way it operates, however, is the basis to understanding how all the other programs operate as well. The first part of the code defines all the variables used throughout the code and also initializes values. In the manual program all variables are either integers (int), long integers (long), and some of the variables have been defined as constants (const int). Integers can store numbers ranging from -2^15 to ((2^15)-1). Long integers can store numbers from -2^31 to ((2^31)-1). The long integers here are used to keep track of time when measured in micro seconds, which involve large numbers that build quickly. The constants are used to define pin modes. For example, pin 12 is defined as the pin which will output the motor step command.

Next the program defines the pin modes.

For example: **pinMode(steppin, OUTPUT);**

The variable steppin has been defined as an integer constant with a value of 12, therefore the above line of code will tell the Arduino that pin 12 will be used as an output.

All of the above code operates in what has been defined as a "void setup" loop, which is a loop that only runs once; just enough to initialize all the pin modes and define the variables. The next chunk of code operates in the "void loop" which runs indefinitely, re-executing the code until the Arduino is manually reset. At the beginning of the loop the Arduino first reads in the inputs which it receives, such as the high/low values from the valve limit switches, as well as a small manual switch which the user can use to define the direction of motor travel.

For example: **dirsel = digitalRead(dirselpin);**

This command stores the high or low signal being received at dirselpin (defined previously in the code as 8) into the variable dirsel. In this case dirsel stands for direction select, therefore the manual direction switch is wired into the Arduino at pin 8. Arduino has built in pull-up resistors, therefore if no input is provided to a pin that has been enabled as a digital input it will read 5V (ie "high") by default. If a user wishes to affect the input, the pin is grounded in order to pull it to 0V (ie "low"). The manual direction switch and the limit switches that indicate the valve is fully open or fully closed are all hooked up such that when they are closed it actually provides ground to the input pins. This may be counter intuitive; as initially it may be thought that when closed the switches would be providing 5V to the input pins.

The next part of code that operates in the "void loop" is a "while loop" which is used to bring the motor back to a home position when it is powered on. The way which the code is setup now is such that home is defined as "valve closed". The Arduino knows that the valve has reached the closed position when the limit switch is activated and grounds the associated input pin on the Arduino board. The "while loop" runs and does not allow the rest of the code in the "void loop" to be executed until the limit switch is

activated AND the manual direction switch is commanding the valve to close. For example, if when the valve is powered on it's already in the open position and the valve direction switch is commanding the valve to open, the Arduino will cause the motor to shut the valve fully (regardless of user input) and will not allow user control until the direction select switch is also commanding the motor to close. This is so that when the code jumps out of the "while loop" the valve does not immediately start opening again. This is a safety precaution. The code within the "while loop" causes the step output pin of the Arduino to oscillate between 0 and 5V at a steady rate and the direction pin to be set low (closing direction). The method used to oscillate the step output is described below.

The step pin is oscillated at a rate defined in the initialization portion of the code. The pin is set high or low using a small "if" statement which is shown below:

```
currentMicrosStep=micros();
if (currentMicrosStep - previousMicrosStep > stepinterval) {
   previousMicrosStep = currentMicrosStep;
   if (motorstep == HIGH)
     motorstep = LOW;
   else
     motorstep = HIGH;
 }
```

The first line assigns the current program time to the variable "currentMicroStep". In the "if" statement this current time is compared to the previous time that was stored and the program determines if the time elapsed has been larger than the user defined "stepinteval". If that amount of time has elapsed then the "if" statement will execute; the first thing done is store the current time as the previous time, that way the "if" statement will know what time the last motor step occurred when it returns on the next loop. Next, if the step pin is currently set to "high" then the program will set it to "low", and vise-versa. You can see that the "stepinterval" is actually half a period, therefore the motor makes one step for every two "stepinterval" increments.

Note that the step pin is oscillating whenever the Arduino has power; the motor is only stopped by using the enable command on the motor controller. For example, when the valve is fully open and the limit switch pulls the associated Arduino pin to 0V, the code sends a high signal to the enable pin which causes the motor controller to lock the step motor in the current position. All the while the step pin continues to oscillate; its output is simply ignored.

Very importantly there is the code which keeps the valve from going beyond 90 degrees open and 0 degrees closed. This is important because if the valve goes much further beyond these bounds it will bind. To prevent this from happening there are the two limit switches that are activated by the valve stem. As described before, whenever a limit switch is activated it grounds a pin on the Arduino board. If the input pin for either limit switch is activated AND the current direction selection is in the direction of that limit switch a signal is sent to the enable port of the motor controller, causing the motor to lock in position. The second condition for motor locking is important, otherwise once the valve hit a limit switch it would be trapped in that position. This operation is built

with a single “if”, “if else”, “else” statement, and is present in both the “while loop” which jogs the motor home on startup as well as the user controlled section of the code.

```
int dir=LOW;
int motorstep=HIGH;
int wait=2000;
int dirsel=LOW;
int opensel=LOW;
int closedsel=LOW;
int ensel=LOW;
int startup=1;
long previousMicrosStep=0;
long previousMillisDir=0;
long joginterval=1200;
long stepinterval=1200;
const int steppin=12;
const int dirpin=11;
const int enpin=13;
const int dirselpin=8;
const int openpin=10;
const int closedpin=9;
const int closingdir=LOW;
const int openingdir=HIGH;

void setup() {

  // initialize variables and assign pin functions
  pinMode(steppin, OUTPUT);
  pinMode(enpin, OUTPUT);
  pinMode(dirpin, OUTPUT);
  pinMode(dirselpin, INPUT);
  pinMode(openpin, INPUT);
  pinMode(closedpin, INPUT);
  digitalWrite(openpin, HIGH);
  digitalWrite(closedpin, HIGH);
  digitalWrite(dirselpin, HIGH);
  delay(wait);
  //Serial.begin(4800);
  //................................//
```

```
}

void loop() {

  unsigned long currentMicrosStep=micros();
   dirsel =digitalRead(dirselpin);
   opensel =digitalRead(openpin);
   closedsel =digitalRead(closedpin);

  //jog valve to closed on startup
  while(startup==1) {
     startup = 1;
    unsigned long currentMicrosStep=micros();
     opensel =digitalRead(openpin);
     closedsel =digitalRead(closedpin);
    dirsel = closingdir;
    if (currentMicrosStep - previousMicrosStep > stepinterval) {
     previousMicrosStep = currentMicrosStep;
      if (motorstep ==HIGH)
         motorstep =LOW;
      else
         motorstep =HIGH;
    }
    if ((opensel==LOW)&&(dirsel==HIGH)) {
      ensel=HIGH;
    }
    else if ((closedsel==LOW)&&(dirsel==LOW)) {
      ensel=HIGH;
    }
    else {
      ensel=LOW;
    }
    digitalWrite(enpin,ensel);
    digitalWrite(dirpin,dirsel);
    digitalWrite(steppin, motorstep);
     dirsel =digitalRead(dirselpin);
    if ((closedsel==LOW)&&(dirsel==LOW)) {
      startup=0;
      ensel=HIGH;
     }
  }
  //................................//


  //Human Motor Control
```

```
  if (currentMicrosStep - previousMicrosStep > stepinterval) {
   previousMicrosStep = currentMicrosStep;
    if (motorstep ==HIGH)
       motorstep =LOW;
    else
       motorstep =HIGH;
  }
  if ((opensel==LOW)&&(dirsel==HIGH)) {
    ensel=HIGH;
  }
  else if ((closedsel==LOW)&&(dirsel==LOW)) {
    ensel=HIGH;
  }
  else {
    ensel=LOW;
  }
  //Serial.print(dirsel);
  //.................................//

  digitalWrite(enpin,ensel);
  digitalWrite(dirpin,dirsel);
  digitalWrite(steppin, motorstep);

}
```

APPENDIX E: SPECIFIC ANGLE ARDUINO CODE

Generating the necessary information to characterize the valve is a large portion of this thesis work, and requires its own program. Characterizing the valve will require repeated runs of the apparatus, all at different valve positions. The valve will be held a constant position for the duration of a flow test, then the tank will be refilled and the valve brought to a different position for a new test. The program is identical to the user controlled program, with the exception of the user defined direction, and an additional condition causes the motor to be locked in position at the desired angle instead of continuing until it reaches a limit switch.

Once the valve has executed the "while loop" that brings it to the home position and the user has flipped the direction control switch to "open", the Arduino begins to count step cycles with a simple variable that increases by one every time the step pin completes a cycle (ie every time the motor makes a step); an additional "if" statement compares the step count to the desired angle, and if the step count is greater than the desired angle it locks the motor position. The user has the ability to enter the desired angle at the beginning of the program, and it then converted into the number of steps using Equation 1.

$$N = \left(\frac{\theta}{1.8} \right) \times 30$$

Equation 1: Angle in Degree to Step Number

The 1.8 pertains to the number of degrees the motor rotates for one step input, and the 30 represents the gear ratio between the motor and the valve stem.

```
int dir=LOW;
int motorstep=HIGH;
int wait=2000;
int dirsel=LOW;
int opensel=LOW;
int closedsel=LOW;
int ensel=LOW;
int startup=1;
long angle=80;
long previousMicrosStep=0;
long previousMillisDir=0;
long joginterval=1200;
long stepinterval=1200;
long stepcount=0;
const int steppin=12;
const int dirpin=11;
const int enpin=13;
const int dirselpin=8;
const int openpin=10;
const int closedpin=9;
const int closingdir=LOW;
const int openingdir=HIGH;

void setup() {

  // initialize variables and assign pin functions
  pinMode(steppin, OUTPUT);
  pinMode(enpin, OUTPUT);
  pinMode(dirpin, OUTPUT);
  pinMode(dirselpin, INPUT);
  pinMode(openpin, INPUT);
  pinMode(closedpin, INPUT);
  digitalWrite(openpin, HIGH);
  digitalWrite(closedpin, HIGH);
  digitalWrite(dirselpin, HIGH);
  angle=(angle/1.8)*30;
  delay(wait);
  //Serial.begin(9600);
  //..............................//
```

```
}

void loop() {

  unsigned long currentMicrosStep=micros();
  dirsel = digitalRead(dirselpin);
  opensel = digitalRead(openpin);
  closedsel = digitalRead(closedpin);

  //jog valve to closed on startup
  while(startup==1) {
    startup = 1;
    unsigned long currentMicrosStep=micros();
    opensel = digitalRead(openpin);
    closedsel = digitalRead(closedpin);
    dirsel = closingdir;
    if (currentMicrosStep - previousMicrosStep > stepinterval) {
      previousMicrosStep = currentMicrosStep;
      if (motorstep == HIGH)
        motorstep = LOW;
      else
        motorstep = HIGH;
    }
    if ((opensel==LOW)&&(dirsel==HIGH)) {
      ensel=HIGH;
    }
    else if ((closedsel==LOW)&&(dirsel==LOW)) {
      ensel=HIGH;
    }
    else {
      ensel=LOW;
    }
    digitalWrite(enpin,ensel);
    digitalWrite(dirpin,dirsel);
    digitalWrite(steppin, motorstep);
    dirsel = digitalRead(dirselpin);
    if ((closedsel==LOW)&&(dirsel==LOW)) {
      startup=0;
      ensel=HIGH;
    }
  }
  //..............................//
```

```
  //Human Motor Control

  if (currentMicrosStep - previousMicrosStep > stepinterval) {
  previousMicrosStep = currentMicrosStep;
    if (motorstep == HIGH) {
      motorstep = LOW;
      if ((startup==0)&&(dirsel==openingdir)) {
        stepcount = stepcount+1;
      }
    }
    else {
      motorstep = HIGH;
    }
  }
  if ((opensel==LOW)&&(dirsel==HIGH)) {
    ensel=HIGH;
  }
  else if ((closedsel==LOW)&&(dirsel==LOW)) {
    ensel=HIGH;
  }
  else {
    ensel=LOW;
  }
  if ((stepcount>angle)&&(dirsel==HIGH)) {
    ensel=HIGH;
  }
  //Serial.print(stepcount);
  //..............................//

  digitalWrite(enpin,ensel);
  digitalWrite(dirpin,dirsel);
  digitalWrite(steppin, motorstep);

}
```

APPENDIX F: CONTROLLER ARDUINO CODE

This is the most complex of the programs, and is used to run the closed loop flow control. Unlike the previous two programs, the Arduino will be looking for input from the load cell as well as generating output for the flow control valve. At the heart, the step commands are sent in the same way as the other two programs, except that the step interval is variable, the same initialization sequence to bring the valve to the home position is also implemented. The variable step size allows the angular velocity of the motor to be adjusted beyond either on/off. The program now also receives an analog input from the load cell. The Arduino then differentiates this mass measurement to get a mass flow reading which can be compared to the reference signal. Depending on the error, an appropriate valve angular velocity is determined, which is then converted into a given step interval. Since the motor controller has a minimum step interval that it can process there are saturation limits placed to constrain how small the step interval can be made.

For the controller program there are some new variables defined, such as KI, and KP proportionality constants. Another new aspect in this program is the type of variables. Now that there is an analog input as well as a derivative calculation integers are not accurate enough; especially if the desired mass flow is only .5 or 1lbm/sec. Therefore the load measurements and the resulting mass flow calculations are all carried out with floating point numbers.

For example: **float previousloadval=0;**

At the beginning of the "void loop" there is the same exact "while loop" which brings the valve to the home position. Before the code reaches the controller portion, the Arduino reads in the value from the load cell using the "analogRead" command. The Arduino has a 10-bit ADC (Analog to digital converter) therefore 0V corresponds to a value of 0, and 5V corresponds to a value of 1023. The output from the load cell is only 0V to .6256V; therefore the load value that the Arduino sees is 0 to 128 approximately. More resolution using a better ADC would be desirable, but at this point the setup is workable.

```
        currentMicrosStep=micros();
    loadval = analogRead(loadpin);
        loadvalfilt=.999*loadvalfilt+.001*loadval;
    SAMPLEINTERVAL=ABS(CURRENTMICROSSTEP-
SAMPLEPREVIOUSMICROSSTEP);
    SAMPLEPREVIOUSMICROSSTEP=CURRENTMICROSSTEP;
    PREVIOUSLOADVAL=CURRENTLOADVAL;
    CURRENTLOADVAL=1.453*LOADVALFILT;
    MDOT=((PREVIOUSLOADVAL-
CURRENTLOADVAL)*1000000)/(SAMPLEINTERVAL);
    ERROR=MASSDOT-MDOT;
    ERRORINT=ERRORINT+(ERROR*SAMPLEINTERVAL/1000000);
    INPUT=(KP*ERROR+KI*ERRORINT);
```

```
STEPINTERVAL=(ABS((1000000*1.8)/((INPUT)*2*30)));
IF (STEPINTERVAL < STEPINTERVALMIN) {
  STEPINTERVAL=STEPINTERVALMIN;
}
```

Shown above is the portion of code that determines the step interval and motor direction based on the load value reading, these lines of code also determines the mass flow based on the derivative of the load value reading. In other words this is the real heart of the program. The other areas of the program regarding the step oscillation are all very similar to the other two programs and will not be covered in much detail in this section.

The first thing that happens is the collection of the load reading and the conversion of the load cell digital reading stored in the "loadval" variable to a number reflecting pounds which is stored in the "currentloadval" variable. Multiplying the digital signal (ie 0-1023) value by 1.453 gives the weight in pounds. This constant is simply derived, as can be seen from Equation 1. The "loadval" variable is actually filtered before being manipulated, since there is some noise from the 10 bit ADC. This filtering is minimal and only consists of a 1000 point moving average. This can be seen in the third line of code above.

$$\frac{186lbf}{128} = 1.453lbf$$

Equation 1: Digital Output to Weight Measurement

Where the 186lb is the weight of the water in the tank when full, and the 128 is the digital output of the load cell when the tank is full.

The next line is where the actual mass flow is determined by approximating the derivative at that moment. This is stored in the variable "mdot". The 1000000 multiplier is because the time interval is in microseconds, and the mass flow is recorded in lbm/sec. Remember that "previousloadval" is the weight of water at the end of the previous sample interval and "currentloadval" is the weight of the water at the end of the current sample interval. The time between these two samplings is, of course, the "sampleinterval" which is measured in microseconds.

Now that an actual mass flow is determined it must be compared to a reference value, which is stored in a variable called "massdot" (not be confused with "mdot" which is the actual mass flow). Subtracting the two values gives the error value that the controller needs to operate. This program does not support a differential controller, only proportional and integral. The error value can be used directly for the proportional controller, but needs to be integrated over time in order to implement the integral controller. This integrated error is called "errorint" and is found by adding the current error value, multiplying it by the sample interval and then adding it to the previous "errorint" value. In this way the "errorint" value keeps getting overridden each time the error is evaluated (ie. once every "sampleinterval").

Alone, the error and integrated error values have little use to determine a system input, next they will need to be multiplied by the appropriate constants (KP and KI) which will determine an angular velocity of the valve. This angular velocity then needs

to be translated into a step interval. The angular velocity is stored in a variable called "input". The equation for which is below:

$$input = KP * error + KI * error \text{ int}$$

Equation 2: Calculating Angular Velocity

This desired angular valve angular velocity must be translated into a step interval for the motor, if the input is small (ie a low angular velocity) then the step interval must be longer, higher angular velocity means a short step interval. The derivation of the equation is shown below.

$$\dot{\phi} = Motor\_RPM = \frac{1.8\left(\frac{\deg}{step}\right)}{step \text{ int}(\sec) * 2}$$

$$\dot{\theta} = \frac{\dot{\phi}}{30\left(\frac{\deg}{\deg}\right)} = \frac{1.8\left(\frac{\deg}{step}\right)}{step \text{ int}(\sec) * 2 * 30\left(\frac{\deg}{\deg}\right)}$$

$$also:$$

$$\dot{\theta} = KP * error + KI * \int error * dt$$

$$\therefore$$

$$KP * error + KI * \int error * dt = \frac{1.8\left(\frac{\deg}{step}\right)}{step \text{ int}(\sec) * 2 * 30\left(\frac{\deg}{\deg}\right)}$$

$$step \text{ int}(\sec) = \frac{1.8\left(\frac{\deg}{step}\right)}{\left(KP * error + KI * \int error * dt\right) * 2 * 30\left(\frac{\deg}{\deg}\right)}$$

$$step \text{ int}(\mu \sec) = 1 * 10^{6} \frac{1.8\left(\frac{\deg}{step}\right)}{\left(KP * error + KI * \int error * dt\right) * 2 * 30\left(\frac{\deg}{\deg}\right)}$$

Equation 3: Determining Step Interval from Valve Angular Velocity

Lastly, there are two "if" statements that determine the direction of the motor depending on whether the "input" (ie valve angular velocity) is positive or negative. The limit switch protection is also still in place to prevent the valve from binding.

```
int dir=LOW;
int motorstep=HIGH;
int dirsel=LOW;
int opensel=LOW;
int closedsel=LOW;
int ensel=LOW;
int startup=1;
int loadval=0;
int joginterval=1200;
int stepintervalmin=1200;
int stepinterval=1200;
int sampleinterval=0;
int initialize=0;
float massdot=2;
float mdot=0;
float KP=20;
float error=0;
float errorint=0;
float KI=20;
float input=0;
float previousloadval=0;
float currentloadval=0;
float loadvalfilt=0;
unsigned long previousMicrosStep=0;
unsigned long currentMicrosStep=0;
unsigned long samplepreviousMicrosStep=0;
unsigned long time=0;

const int loadpin=A0;
const int steppin=12;
const int dirpin=11;
const int enpin=13;
const int dirselpin=8;
const int openpin=10;
const int closedpin=9;
const int closingdir=LOW;
const int openingdir=HIGH;
```

```
void setup() {

  // initialize variables and assign pin functions
  pinMode(steppin, OUTPUT);
  pinMode(enpin, OUTPUT);
  pinMode(dirpin, OUTPUT);
  pinMode(dirselpin, INPUT);
  pinMode(openpin, INPUT);
  pinMode(closedpin, INPUT);
  digitalWrite(openpin, HIGH);
  digitalWrite(closedpin, HIGH);
  digitalWrite(dirselpin, HIGH);
//  Serial.begin(4800);
  //................................//

}

void loop() {

  currentMicrosStep=micros();
  dirsel = digitalRead(dirselpin);
  opensel = digitalRead(openpin);
  closedsel = digitalRead(closedpin);
  loadval = analogRead(loadpin);
  loadvalfilt=.999*loadvalfilt+.001*loadval;

  //jog valve to closed on startup
  while(startup==1) {
    startup = 1;
    currentMicrosStep=micros();
    opensel = digitalRead(openpin);
    closedsel = digitalRead(closedpin);
    dirsel = closingdir;
    if (currentMicrosStep - previousMicrosStep > stepintervalmin) {
      previousMicrosStep = currentMicrosStep;
      if (motorstep == HIGH)
        motorstep = LOW;
      else
        motorstep = HIGH;
    }
    if ((opensel==LOW)&&(dirsel==openingdir)) {
      ensel=HIGH;
    }
```

```
  else if ((closedsel==LOW)&&(dirsel==closingdir)) {
    ensel=HIGH;
  }
  else {
    ensel=LOW;
  }
  digitalWrite(enpin,ensel);
  digitalWrite(dirpin,dirsel);
  digitalWrite(steppin, motorstep);
  dirsel = digitalRead(dirselpin);
  if ((closedsel==LOW)&&(dirsel==LOW)) {
    startup=0;
    ensel=HIGH;
  }
}
//..............................//


//Controller Active
if (dirsel==closingdir){
  stepinterval=stepintervalmin;
}

else {
  sampleinterval=abs(currentMicrosStep-samplepreviousMicrosStep);
  samplepreviousMicrosStep=currentMicrosStep;
  previousloadval=currentloadval;
  currentloadval=1.453*loadvalfilt;
  mdot=((previousloadval-currentloadval)*1000000)/(sampleinterval);
  error=massdot-mdot;
  errorint=errorint+(error*sampleinterval/1000000);
  input=(KP*error+KI*errorint);
  stepinterval=(abs((1000000*1.8)/((input)*2*30)));
  if (stepinterval < stepintervalmin) {
    stepinterval=stepintervalmin;
  }
  if (initialize==0){
    error=0;
    errorint=0;
    initialize=1;
  }
  //Serial.println(time);
  //Serial.println(currentloadval);
  //Serial.println(sampleinterval);
  //Serial.println(99);
```

```
    if (input >= 1) {
      dirsel=openingdir;
    }
    if (input < 1) {
      dirsel=closingdir;
    }
  }
  if (currentMicrosStep - previousMicrosStep > stepinterval) {
    previousMicrosStep = currentMicrosStep;
    if (motorstep == HIGH)
      motorstep = LOW;
    else
      motorstep = HIGH;
  }
  if ((opensel==LOW)&&(dirsel==openingdir)) {
    ensel=HIGH;
  }
  else if ((closedsel==LOW)&&(dirsel==closingdir)) {
    ensel=HIGH;
  }
  else {
    ensel=LOW;
  }

/  Serial.print(mdot);
/  Serial.print(99);
  //..............................//

  digitalWrite(enpin,ensel);
  digitalWrite(dirpin,dirsel);
  digitalWrite(steppin, motorstep);
```

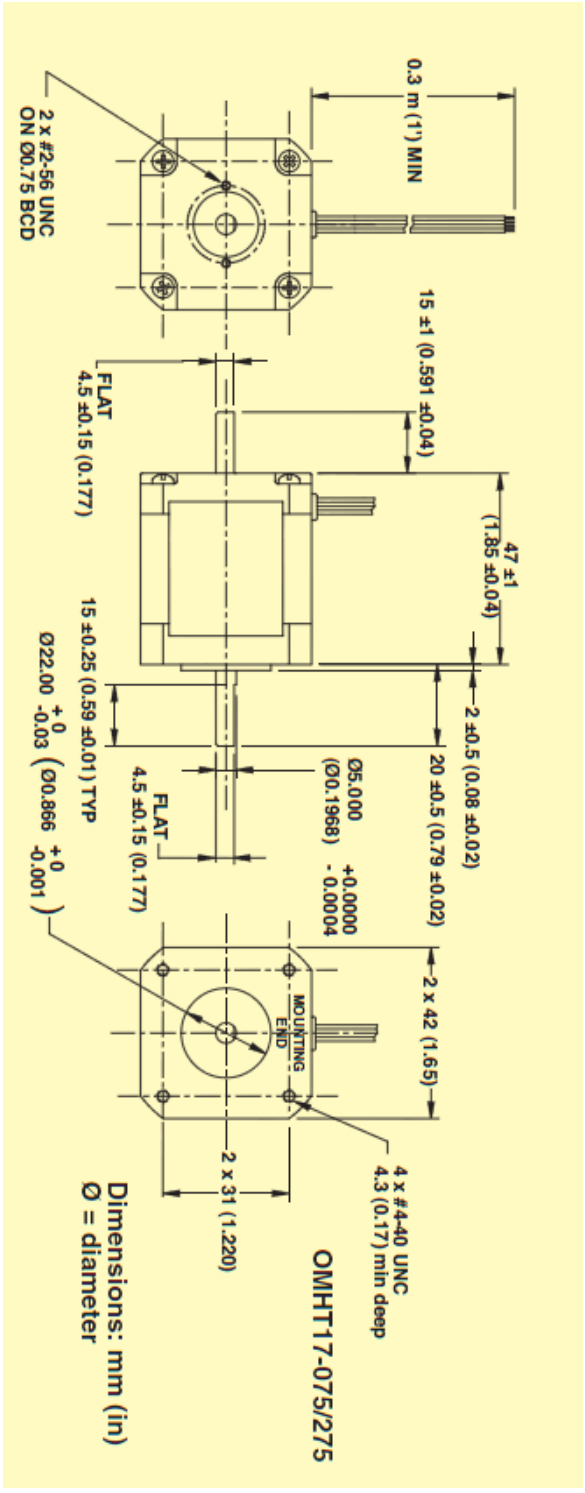
APPENDIX G: TECHNICAL DATA SHEETS

Step Motor:

RECOMMENDED MOTORS

MODEL NO.	MOTOR CONNECTION 1 = SERIES 2 = PARALLEL 3 = UNIPOLAR	MOTOR LENGTH mm (in)	MAXIMUM HOLDING TORQUE[2] (oz-in)	LEADS	STEP ANGLE (DEG)	VOLTS	AMPS	OHMS	MH	ROTOR INERTIA (oz-in²/ g-cm²)	MOTOR WEIGHT g (lb)
OMHT11-013	2	48 (1.87)	15	4	1.8	2.0	1.0	2.0	2.6	0.098/18	177 (0.39)
OM5014-842	2	40 (1.57)	26.0	4	1.8	4.8	1.0	4.3	5.5	0.109/20	213 (0.47)
OMHT17-075	1	47 (1.85)	62.8	8	1.8	5.7	0.85	6.6	12.0	0.37/68	331 (0.73)
	2					2.8	1.70	1.7	3.0		
	3		44.4			4.0	1.20	3.3	3.0		
OMHT17-275	1	48 (1.90)	62.3	8	1.8	5.7	0.85	6.6	10.0	0.44/82	357 (0.79)
	2					2.8	1.70	1.7	2.5		
	3		44.0			4.0	1.20	3.3	2.5		
OMHT23-393	1	39 (1.54)	76.6	8	1.8	7.4	0.71	1.7	21.6	0.66/120	454 (1.00)
	2					3.7	1.41	2.6	5.4		
	3		54.2			5.2	1.00	5.2	5.4		
OMHT23-593	1	41 (1.61)	79.3	8	1.8	7.4	0.71	10.4	26.1	0.73/135	417 (0.92)
	2					3.7	1.41	2.6	6.6		
	3		59.4			5.2	1.00	5.2	6.6		
OMHT23-397	1	54 (2.13)	177.0	8	1.8	5.1	1.41	3.6	10.0	1.64/300	699 (1.54)
	2					2.5	2.83	0.9	2.5		
	3		125.0			3.6	2.00	1.8	2.5		
OMHT23-597	1	54 (2.13)	177.0	8	1.8	5.1	1.41	3.6	10.8	1.42/260	599 (1.32)
	2					2.5	2.83	0.9	2.7		
	3		127.4			3.6	2.00	1.8	2.7		
OMHT23-400	1	76 (2.99)	264.0	8	1.8	6.4	1.41	4.5	14.4	2.62/480	998 (2.20)
	2					3.2	2.83	1.1	3.6		
	3		187.0			4.5	2.00	2.3	3.6		
OMHT23-600	1	76 (2.99)	264.8	8	1.8	6.4	1.41	4.5	15.6	2.51/460	998 (2.20)
	2					3.2	2.83	1.1	3.9		
	3		187.0			4.5	2.00	2.3	3.9		
OMHT23-550[1]	1	78 (3.05)	255.0	8	1.8	6.3	1.41	4.5	15.2	2.62/480	998 (2.20)
	2					3.2	2.83	1.13	3.8		
	3		180.5			4.5	2.00	2.25	3.8		
OMHT24-100	2	44 (1.73)	123	4	1.8	2.0	2.8	0.73	1.6	1.42/260	599 (1.32)
OMHT24-105	2	54 (2.13)	177	4	1.8	1.7	4.0	0.43	1.1	2.46/450	830 (1.83)
OMHT24-108	2	85 (3.35)	354	4	1.8	2.6	4.0	0.65	2.4	4.91/900	1402 (3.09)
OMHT34-504	1	66 (2.62)	396	8	1.8	3.05	3.18	0.96	6.8	6.0/1100	1588 (3.5)
	2					1.51	6.3	0.24	1.7		
	3		297			2.16	4.5	0.48	1.7		
OMHT34-485	1	79 (3.11)	650	8	1.8	3.2	4.3	0.76	5.2	7.8/1400	2803 (6.18)
	2					1.6	8.6	0.19	1.3		
	3		455			2.26	6.0	0.38	1.3		
OMHT34-505	1	96 (3.78)	849	8	1.8	4.20	3.18	1.32	10.8	10.1/1850	2676 (5.9)
	2					2.08	6.3	0.33	2.7		
	3		608			2.97	4.5	0.66	2.7		
OMHT34-486	1	118 (4.63)	1200	8	1.8	4.4	4.1	1.08	8.8	14.6/2680	3810 (8.40)
	2					2.2	8.1	0.27	2.2		
	3		840			3.1	5.7	0.54	2.2		
OMHT34-506	1	125 (4.94)	1260	8	1.8	5.43	2.8	1.94	21.6	15.0/2750	3810 (8.4)
	2					2.74	5.6	0.49	5.4		
	3		906			3.88	4.0	0.97	5.4		

D-5


0.3 m (1') MIN
15 ±1 (0.591 ±0.04)
47 ±1
(1.85 ±0.04)
2 ±0.5 (0.08 ±0.02)
20 ±0.5 (0.79 ±0.02)
2 x 42 (1.65)
2 x 31 (1.220)
2 x #2-56 UNC
ON Ø0.75 BCD
FLAT
4.5 ±0.15 (0.177)
15 ±0.25 (0.59 ±0.01) TYP
Ø5.000
(Ø0.1968) +0.0000 - 0.0004
FLAT
4.5 ±0.15 (0.177)
Ø22.00 +0 -0.03 (Ø0.866 +0 -0.001)
MOUNTING END
4 x #4-40 UNC
4.3 (0.17) min deep
OMHT17-075/275
Dimensions: mm (in)
Ø = diameter

Gearbox:

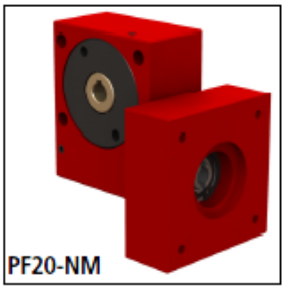
GEARBOXES

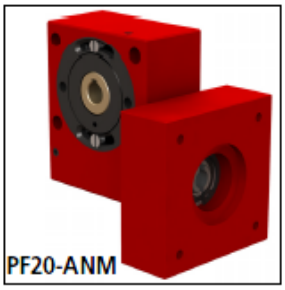
Precision Worm Gear Reducers

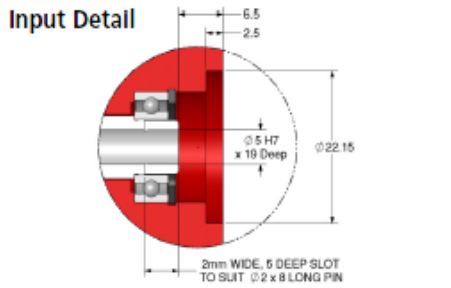
Nema 17 Flange Input 1.25Nm – 5Nm : 10:1 - 120:1 Ratio

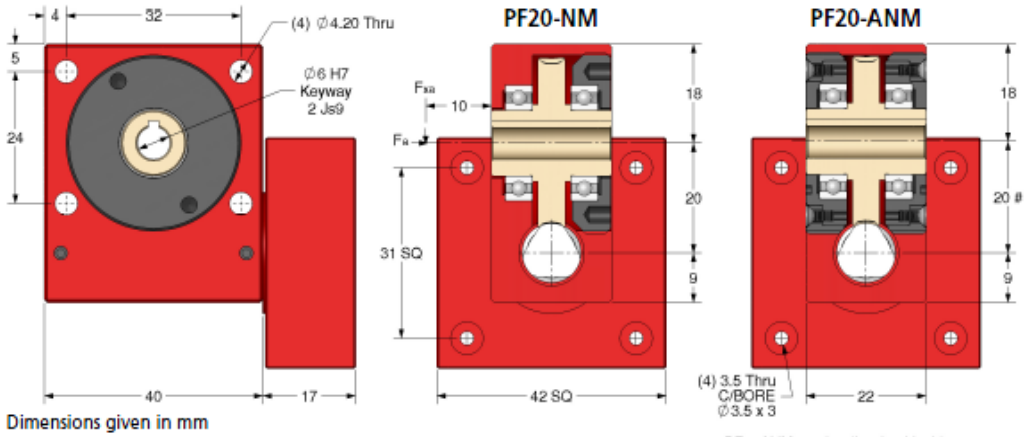
e-cad Drawings Available

PF20NM









Dimensions given in mm

PF20ANM version (low backlash) output is on eccentric so centres will vary.

Part Number Standard ≤30'	Part Number Low Backlash ≤8'	Ratio	Efficiency at 1000 Rpm	Reflected Inertia at Input ($kg.m^2$)	Self Locking Output
PF20-10NM	PF20-10ANM	10:1	86%	2.53×10^{-7}	✖
PF20-12NM	PF20-12ANM	12:1	85%	2.43×10^{-7}	✖
PF20-15NM	PF20-15ANM	15:1	84%	2.35×10^{-7}	✖
~~PF20-20NM~~	~~PF20-20ANM~~	~~20:1~~	~~78%~~	~~2.28×10^{-7}~~	✖
PF20-30NM	PF20-30ANM	30:1	71%	2.24×10^{-7}	✖
PF20-60NM	PF20-60ANM	60:1	60%	2.21×10^{-7}	✔
PF20-120NM	PF20-120ANM	120:1	32%	2.00×10^{-7}	✔
PF20-SPNM	PF20-SPANM	5:1-120:1	Special Ratios: Replace **SP** with required ratio		
Accessories					
P20-X	P20-DX	Single (X) & Double (DX) Output Shafts			
KK2-22		Output Shaft Key			

Weight: 0.50 lb. **Greased for life:** Shell Gadus S5 V42P 2.5.
F_{xa} at 1000rpm: 26.5 lb. **F_a at 1000rpm:** 11.0 lb.

Modifications and custom designed gearboxes available. Contact Sales for details.
Testing in your application is necessary.
You will need to assess duty cycles and confirm suitability with your own calculations.
All figures listed are to be used for guidance only.

Output Torque Nm

Rpm Input	Reduction Ratio 10:1	12:1	15:1	20:1	30:1	60:1	120:1
3000	1.40	1.50	1.60	1.70	1.90	2.10	1.25
2000	1.70	1.70	1.80	1.80	2.10	2.40	1.42
1000	2.10	2.30	2.30	2.50	2.70	2.90	1.70
500	2.70	2.70	2.80	2.90	3.20	3.50	1.94
200	3.40	3.50	3.50	3.70	3.90	4.10	2.27
100	3.90	3.90	4.10	4.20	4.30	4.50	2.55
50	4.40	4.50	4.50	4.60	4.70	4.80	2.72
10	4.80	4.80	4.90	4.90	5.00	5.00	2.72

rino Mechanical Inc.
☎ 516-771-6777
516-771-6444
sales@rinomechanical.com
www.rinomechanical.com

Load Cell:

http://www.massload.com

GN200 S-Type Loadcell

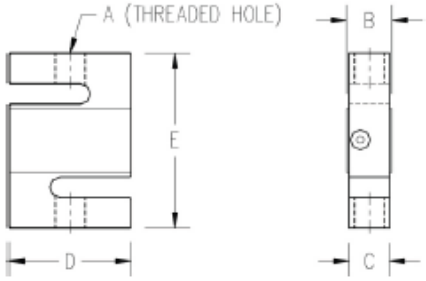
250 lbs (125 kg)
500 lbs (225 kg)
750 lbs (375 kg)
1,000 lbs (450 kg)
1,500 lbs (680 kg)
2,000 lbs (900 kg)
2,500 lbs (1100 kg)
5,000 lbs (2250 kg)
10,000 lbs (4500 kg)
15,000 lbs (6800 kg)

Performance Specifications

Safe Overload	150%
Full Scale Output	3mv/V +/- .25% FS
Maximum Excitation	15 VDC
Non-Linearity	<0.03% FS
Hysteresis	<0.02%FS
Non-Repeatability	<0.01% FS
Thermal Sensitivity Shift	0.0008% of reading/°F
Thermal Zero Shift	0.0015% FS/°F
Barometric Effect	NIL
Bridge Resistance	350 Ω
Operating Temp Range	-50°C to +80°C
Compensated Temp Range	-10°C to +40°C
Sealing Standard	IP-67

The GN200 loadcell is ideally suited to many applications such as scale conversions, tension applications and general purpose weighing.

This versatile, low cost loadcell is available in stainless or nickel plated alloy steel. With a wide range of capacities available, you will find one to fit your needs.



Capacity	A	B		C		D		E	
1K=1000lb	in	in	mm	in	mm	in	mm	in	mm
250lb	1/4"-28 UNF	0.75	19	0.50	13	2.00	51	3.00	76
500lb, 750lb	1/2"-20 UNF	1.00	25	0.75	19	2.00	51	3.00	76
1K, 1500lb	1/2"-20 UNF	1.00	25	0.75	19	2.00	51	3.00	76
2K, 2500lb	1/2"-20 UNF	1.25	32	1.00	25	2.00	51	3.00	76
3K, 5K, 10K	3/4"-16 UNF	1.25	32	1.00	25	3.00	76	4.25	108
15K	1"-14 UNF	1.50	38	1.25	32	3.50	89	5.50	140

301-47th Street East • Saskatoon, SK, Canada • PH: 1-800-667-3825 • Fax: (306) 931-1991

Massload

57

Pressure Transducer:

HOW TO ORDER PX309 SERIES
RUGGED, GENERAL PURPOSE TRANSDUCERS WITH 0 TO 5 Vdc OUTPUT

See Section Y for a Selection of Scientific, Technical, and Reference Books Available from omega.com

Twist-lock style.

PX329-015G5V, $275, shown larger than actual size.

Metric thread adaptors available, see section C.

PX309 Series
0 to 5 Vdc Output
0-1 to 0-10,000 psi
0-70 mbar to 0-690 bar

Starts at $225

NIST CE RoHS
Standard

- Gage or Absolute Pressure
- Low Pressure to 1 psig
- Rugged Solid State Design
- All Stainless Steel Construction
- High Stability, Low Drift
- 0.25% Static Accuracy

5 V Output Specifications

Excitation: 9 to 30 Vdc (reverse polarity and overvoltage protected)

Output: 0 to 5 Vdc

Static Accuracy 5 to 10,000 psi: ±0.25% FS BSL at 25°C; includes linearity, hysteresis and repeatability

Zero Offset: ±2% FSO; ±4% for 1 and 2 psi ranges

Span Setting: ±2% FSO; ±4% for 1 and 2 psi ranges

Compensated Temperature:
- >5 psi Range: -20 to 85°C (-4 to 185°F)
- ≤5 psi Range: 0 to 50°C (32 to 122°F)

Total Error Band: ±2% FSO; includes linearity, hysteresis, repeatability, thermal hysteresis and thermal errors (except 2 psi = ±3% and 1 psi = ±4.5%)

LOW-PRESSURE RANGES HIGHLIGHTED

To Order (Specify Model Number)

RANGE bar	RANGE psi	1.5 m CABLE CONNECTION	PRICE	MINI DIN CONNECTION	PRICE	TWIST-LOCK CONNECTION	PRICE
ABSOLUTE PRESSURE							
0 to 0.34	0 to 5	PX309-005A5V	$325	PX319-005A5V	$325	PX329-005A5V	$350
0 to 1	0 to 15	PX309-015A5V	245	PX319-015A5V	245	PX329-015A5V	295
0 to 2.1	0 to 30	PX309-030A5V	245	PX319-030A5V	245	PX329-030A5V	295
0 to 3.4	0 to 50	PX309-050A5V	245	PX319-050A5V	245	PX329-050A5V	295
0 to 6.9	0 to 100	PX309-100A5V	245	PX319-100A5V	245	PX329-100A5V	295
0 to 14	0 to 200	PX309-200A5V	245	PX319-200A5V	245	PX329-200A5V	295
0 to 21	0 to 300	PX309-300A5V	245	PX319-300A5V	245	PX329-300A5V	295
GAGE PRESSURE							
0 to 0.07	0 to 1	PX309-001G5V	$345	PX319-001G5V	$345	PX329-001G5V	$370
0 to 0.14	0 to 2	PX309-002G5V	325	PX319-002G5V	325	PX329-002G5V	350
0 to 0.34	0 to 5	PX309-005G5V	300	PX319-005G5V	300	PX329-005G5V	325
0 to 1	0 to 15	PX309-015G5V	225	PX319-015G5V	225	PX329-015G5V	275
0 to 2.1	0 to 30	PX309-030G5V	225	PX319-030G5V	225	PX329-030G5V	275
0 to 3.4	0 to 50	PX309-050G5V	225	PX319-050G5V	225	PX329-050G5V	275
0 to 6.9	0 to 100	PX309-100G5V	225	PX319-100G5V	225	PX329-100G5V	275
0 to 10	0 to 150	PX309-150G5V	225	PX319-150G5V	225	PX329-150G5V	275
0 to 14	0 to 200	PX309-200G5V	225	PX319-200G5V	225	PX329-200G5V	275
0 to 21	0 to 300	PX309-300G5V	225	PX319-300G5V	225	PX329-300G5V	275
0 to 34	0 to 500	PX309-500G5V	225	PX319-500G5V	225	PX329-500G5V	275
0 to 69	0 to 1000	PX309-1KG5V	225	PX319-1KG5V	225	PX329-1KG5V	275
0 to 138	0 to 2000	PX309-2KG5V	225	PX319-2KG5V	225	PX329-2KG5V	275
0 to 207	0 to 3000	PX309-3KG5V	225	PX319-3KG5V	225	PX329-3KG5V	275
0 to 345	0 to 5000	PX309-5KG5V	225	PX319-5KG5V	225	PX329-5KG5V	275
0 to 517	0 to 7500	PX309-7.5KG5V	225	PX319-7.5KG5V	225	PX329-7.5KG5V	275
0 to 690	0 to 10,000	PX309-10KG5V	225	PX319-10KG5V	225	PX329-10KG5V	275

Comes complete with 5-point NIST-traceable calibration.

Notes: *1. Units 100 psig and above may be subjected to vacuum on the pressure port without damage.*
2. For alternative performance specifications to suit your application, contact Engineering.

Ordering Examples: PX309-100G5V, *100 psi gage pressure transducer with 0 to 5 Vdc output and 1.5 m cable termination,* ***$225.***

PX319-015A5V, *15 psi absolute pressure transducer with 0 to 5 Vdc output and mini DIN termination,* ***$245.***
PX329-3KG5V, *3000 psi gage pressure transducer with 0 to 5 Vdc output and twist-lock termination,* ***$275.***
Mating connector sold separately; order ***PT06V-10-6S, $26.50.*** *Consult Sales for OEM pricing.*

ACCESSORIES

MODEL NO.	PRICE	DESCRIPTION
CAL-3	$150.00	Recalibration: 5-point NIST traceable
PT06V-10-6S	26.50	Mating connector for PX329
CA-39-4PC22-5	90.00	4-conductor mating twist-lock connector with 1.5 m (5') cable for PX329
CX5302	15.00	Extra mini DIN connector for PX319

Injector:

SS

Small Droplet Size Dense Fog

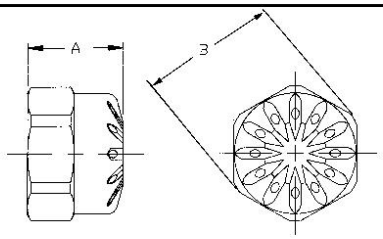
Metal

DESIGN FEATURES
- Multiple flat fan patterns
- Solid one-piece construction
- Female connection

SPRAY CHARACTERISTICS
- Relatively small droplets

Spray pattern: Dense Full Cone
Flow rates: 9.16 to 618 l/min
Spray angles:
SS4.8 thru **SS25** - 35°
SS35 thru **SS70** - 45°

Fog



Dimensions are approximate. Check with BETE for critical dimension applications.

SPECIAL PURPOSE

TO ORDER: specify pipe size, connection type, nozzle number, spray angle, and material.

SS Flow Rates and Dimensions
Full Cone, 3/4", 1" and 1-1/4" Pipe Size, BSP or NPT

Female Pipe Size	Nozzle Number	K Factor	LITERS PER MINUTE @ BAR 0.7 bar	1 bar	2 bar	3 bar	5 bar	10 bar	15 bar	Dimensions (mm) A	B	Wt. (g)
3/4	SS4.8	10.9	9.16	10.9	15.5	19.0	24.5	34.6	42.4	25.4	35.1	85.1
	SS9	20.5	17.2	20.5	29.0	35.6	45.9	64.9	79.5			
	SS12	27.4	22.9	27.4	38.7	47.4	61.2	86.5	106			
	SS18	41.1	34.3	41.1	58.1	71.1	91.8	130	159			
1	SS25	57.0	47.7	57.0	80.6	98.8	127	180	221	26.5	42.2	142
	SS35	70.8	66.8	70.8	113	138	178	252	309			
1 1/4	SS50	114	95.4	114	161	198	255	361	442	31.0	53.1	227
	SS70	160	134	160	226	277	357	505	618			

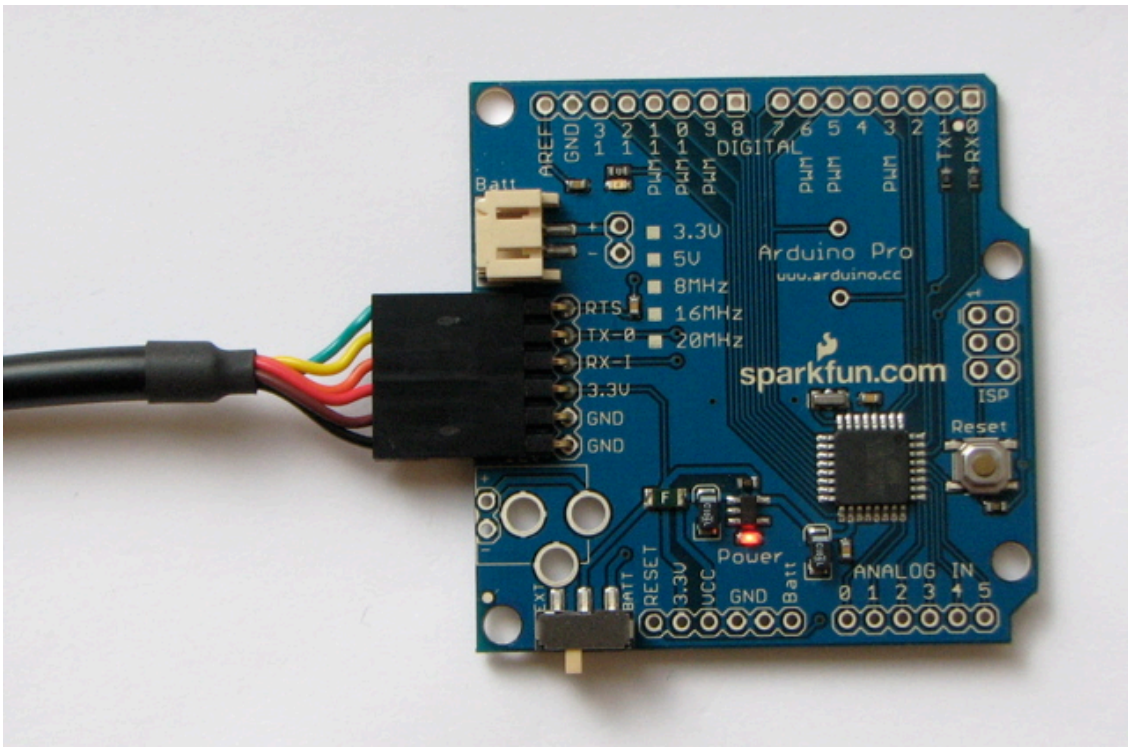
Flow Rate (l/min) = $K\sqrt{bar}$

Standard Materials: Brass, 303 and 316 Stainless Steel.

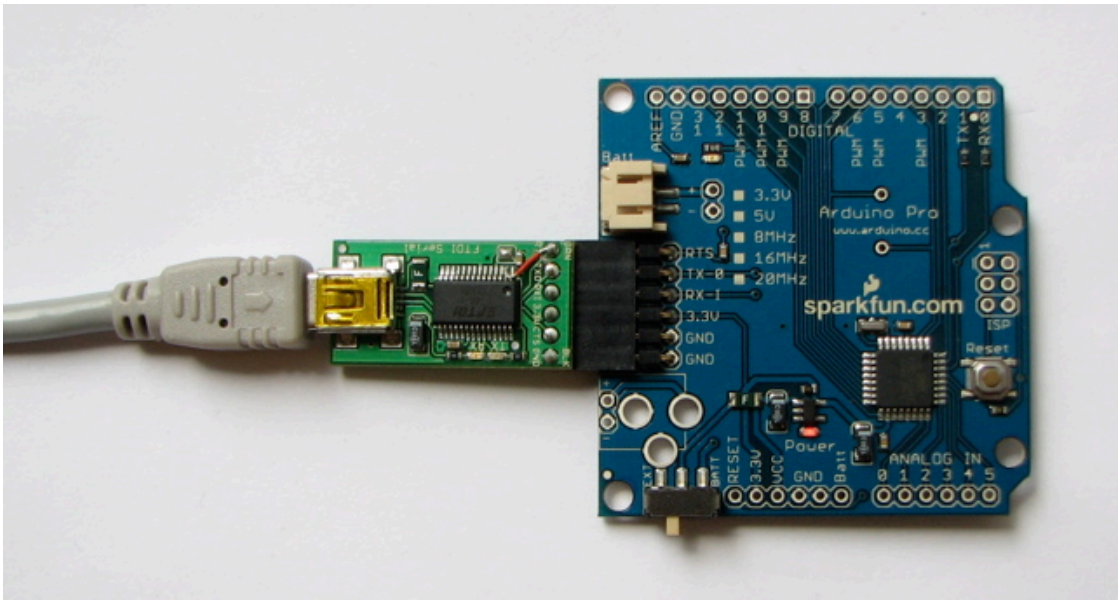
Arduino:

Arduino Pro

The Arduino Pro is intended for advanced users who require flexibility and low-cost. It comes with the minimum of components (no on-board USB or pin headers) to keep the cost down. It's a good choice for a board you want to leave embedded in a project. Please note that there are multiple variants of the board which operate at different voltages and clock speeds. You need to know if you have the 3.3V / 8 MHz version or the 5V / 16 MHz version. The board comes without built-in USB circuitry, so an off-board USB-to-TTL serial convertor must be used to upload sketches. For the 3.3V Arduino Pro boards, this can be a FTDI TTL-232R-3V3 USB - TTL Level Serial Converter Cable or the SparkFun FTDI Basic Breakout Board (3.3V). For the 5V Arduino Pro boards, use a TTL-232R USB - TTL Level Serial Converter or the SparkFun FTDI Basic Breakout Board (5V). (You can probably also get away with using a 5V USB-to-serial convertor with a 3.3V board and vice-versa, but it's not recommended.)
If using the FTDI cable on Windows, you'll need to make one configuration change to enable the auto-reset. With the board connected, open the Device Manager (in Control Panels > System > Hardware), and find the USB Serial Port under Ports. Right-click and select properties, then go to Port Settings > Advanced and check Set RTS on Close under Miscellaneous Options.
For the 3.3V versions of the Arduino Pro, select **Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328**or **Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168** from the **Tools > Board** menu (depending on the microcontroller on your board). For the 5V versions of the Arduino Pro, select **Arduino Duemilanove or Nano w/ ATmega328** or **Arduino Diecimila, Duemilanove, or Nano w/ATmega168**.

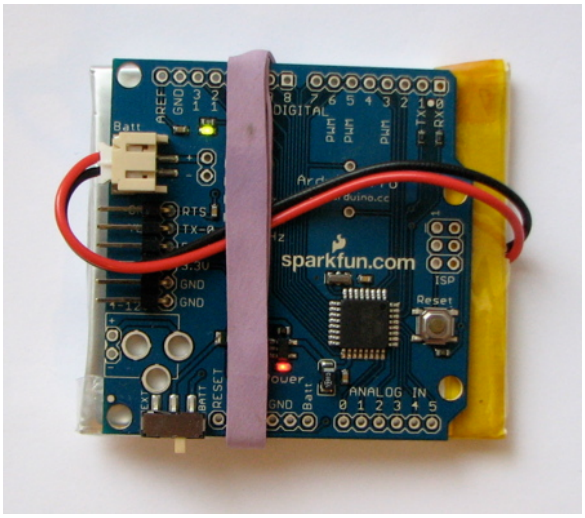


An Arduino Pro connected to (and powered by) an FTDI USB - TTL Level Serial Converter Cable. The green and yellow wires align with the words "green" and "yellow" written underneath the pins.



The Arduino Pro connected to (and powered by) a** SparkFun **FTDI Basic Breakout Board (prototype version) and USB Mini-B cable.

The external USB-to-TTL serial convertor will power the Arduino Pro, regardless of the position of the switch. To use the board standalone, with no connection to a computer, it can be be powered by a battery or an external power supply (wall wart). You can solder the + and - wires of a battery connector to the corresponding holes on the board. For the 3.3V boards, you can connect a LiPo battery (with JST connector) to the JST jack. Alternatively, solder a DC power jack into the three large holes on the board, and connect a DC power supply (center positive). When the switch is in the "Batt" position, the board will draw power from an attached battery; when it is in the "Ext." position, power comes from an external power supply. In either position, the board can be powered by the 6-pin USB header.



***A 3.3V Arduino Pro powered by a** 2000 mAh LiPo battery **from** SparkFun.*

Any standard 0.1" spaced header can be soldered to the holes on the Arduino Pro. To use every pin requires two 6-pin header and two 8-pin headers. Bare wire can also be soldered directly to the holes. Note that the header spacing is compatible with Arduino shields.

Step Motor Controller:

STEPPER MOTOR DRIVE

PULSE AND DIRECTION, FULL AND HALF STEP 2.0 A, 35 VDC

Applied Motion Products
motors • drives • controls

NEW

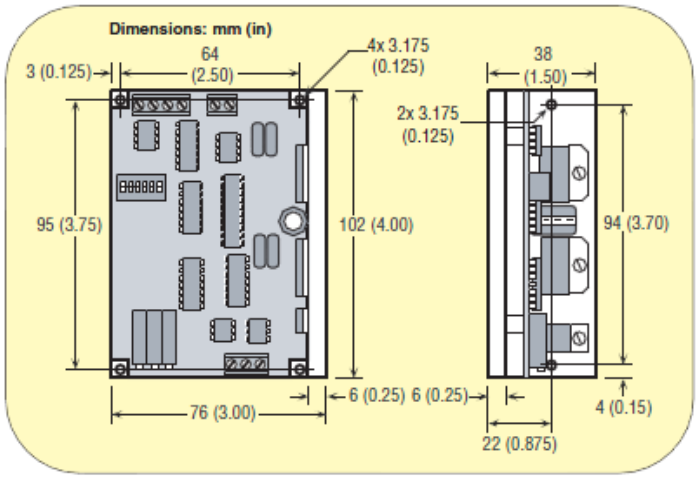
2035 $189

MADE IN USA CE

2035, $189, close to actual size.

- Wide Range of Motor Sizes
- 70 W of Usable Power
- Pulse Width Modulation Switching Amplifiers
- DC Bus Voltage 12 to 35 Vdc Motor Supply (Including Ripple)
- Phase Current from 0.125 to 2.0 A (Switch Selected, 16 Settings)
- Step, Direction and Enable Inputs, Optically Isolated, 5 to 24V
- Inputs Can be Sourcing (PNP) or Sinking (NPN) Type
- Full and Half Step (Switch Selected)
- Automatic 50% Idle Current Reduction (Switch Selected)
- Compact Size: 38.1 x 76.2 x 101.6 mm (1.5 x 3.0 x 4.0")
- CE Compliant

The 2035 step motor driver is a full or half step phase sequencer with three state switching amplifiers and optoisolated circuits. The drive also includes an automatic feature to lower motor current by 50% anytime the motor is left at rest for more than one second. This feature can be disabled. The amplifier regulates motor current by chopping at a constant, inaudible frequency. Phase current is selected from 16 levels by a DIP switch. Full or half step resolution is switch selectable.



SPECIFICATIONS

Amplifiers: Dual, bipolar H-bridge, pulse width modulated switching at 20 KHz. 12 to 35 Vdc input. 0.125 to 2.0 A/phase output current, switch selectable in 0.125 A increments. 70 W maximum output power. Automatic idle current reduction, reduces current to 50% of setting after one second.

Inputs: Step, direction and enable, optically isolated, 5 to 24V logic. 2200Ω input impedance. Motor steps when STEP input turns off. 10 µsec minimum low pulse. 50 µsec minimum set up time for direction signal.

Physical: Mounted on ¼ inch thick black anodized aluminum heat transfer chassis. 38.1 x 76.2 x 101.6 mm (1.5 x 3.0 x 4.0") overall. Power on red LED.

Maximum Chassis Temperature: 70°C (158°F)
Weight: 250 g (9 oz)
Ambient Temp Range (Operating): 0 to 70°C (32 to 158°F)
Connectors: European style screw terminal blocks
Power Supply: 2 position
Motor: 4 position
Signal Input: 4 position
Max Wire Size: 16 AWG

D-14

Step Motor Power Supply:

NEW

POWER SUPPLIES
FOR OPEN FRAME STEPPER DRIVES

OMPS Series Starts at $180

2 YEAR WARRANTY

RoHS

Applied Motion Products
motors • drives • controls

OMPS150A24, $180, shown smaller than actual size.

OMPS300A48, $266, shown smaller than actual size.

OMPS150A24, OMPS300A48

- Universal Input Voltage Range from 85 to 265 Vac
- Built-In Active PFC Filter, PF>0.95, Conforms to EN61000-3-2
- Pending for Safety Approvals: CE, CCC, UL/CSA/EN60950
- EMI: Conform to EN55011-B, EN55022-B, FCC-B
- EMS: Conform to EN61000-4-2, 3, 4, 5, 6, 8, 11
- LED Power Good Indicator
- Peak Current for Motor Application
- 100% Full Load Burn-In Test, High Performance, High Reliability
- Compact Size, 15% Smaller than Conventional Products

OMPS150A24

- 24V Output, Manually Adjustable Output Voltage
- High Flexibility with 3 Optional Connection Methods for Input/Output: Horizontal or Vertical Terminal Block or Connectors to Fit Wire Harness

OMPS300A48

- 48V Output, Manually Adjustable Output Voltage
- High Flexibility with 2 Optional Connection Methods for Input/Output: Horizontal or Vertical Terminal Block
- Fan Speed Control by Output Current to Extend Lifetime
- Remote Sense to Compensate Wire/Connection Voltage Drop

Open-frame stepper drives require a separate DC power supply for operation. Omegamation offers two power supplies that are matched for use with our open-frame stepper drive selection: the OMPS150A24 and the OMPS300A48.

Both power supplies are switched-mode, regulated DC power supplies with active PFC filters. The OMPS150A24 is 24 Vdc, 6.3 A (150 W) while the OMPS300A48 is 48 Vdc, 6.7 A (300 W).

OMPS150A24 SPECIFICATIONS

Nominal Output Voltage: 24V
Maximum Output Current: 6.3 A
Peak Output Current: 9.5 A
Maximum Output Power: 151.2 W
Efficiency (Typical) (115/230 Vac)[1]: 82/85%
Input Voltage Range: 85 to 265 Vac (47 to 63 Hz) or 120 to 370 Vdc
Input Current (Typical) (115/230 Vac)[1]: 1.8/0.9 A
Inrush Current (Typical): 16 A at 115 Vac, 32 A at 230 Vac, Ta = 25°C (77°F), cold start
Harmonic Current: Compliance to EN61000-3-2
Power Factor (Typical) (115/230 Vac)[1]: 0.99/0.95
Output Voltage Range: 21.6 to 26.4V
Ripple and Noise (115/230 Vac)[1, 2]: 150 mV
Line Regulation[2]: 96 mV
Load Regulation[2]: 120 mV
Temperature Coefficient: Less than 0.02 %/°C
Over Current Protection[3]: 6.6/9.7 A
Over Voltage Protection[4]: 27.6 to 32.4V
Hold-Up Time (Typical) (115/230 Vac)[1]: 20 ms
Leakage Current: 0.75 mA max, 0.25 mA (Typical) at 115 Vac, 0.5 mA (Typical) at 230 Vac
Series Operation: Possible
Remote ON/OFF Control: Option, CN3: 4 to 10V, Power ON; 0 to 0.8V, Power OFF
Operating Temperature: -10 to 70°C (14 to 158°F)
Operating Humidity: 20 to 90% RH (no dewdrop)
Storage Temperature: -30 to 85°C (-22 to 185°F)
Storage Humidity: 10 to 95% RH (no dewdrop)
Cooling Method: Convection cooling/ forced air cooling
Withstand Voltage:
Input - Output: 3.0 KVac (20 mA)
Input - FG: 2.0 KVac (20 mA)
Output - FG: 500 Vac (100 mA) for 1 min
Isolation Resistance: More than 100 MΩ at Ta = 25°C (77°F) and 70% RH, Output - FG: 500 Vdc

D-37

DAQ Power Supply

Quad 020V @ 2.5A 3 Fixed

Model: XP581A

Brand: Elenco®

Manufacturer: Elenco®

List Price: $124.95

Quad 0-20V @ 2.5A 3 Fixed

Features:

- This unit will work with an input of 110 OR 220 VAC! (plug adapter may be required)
- 4 DC Voltages:
- 3 Fixed:
- +5V @ 3A
- +12V @ 1A
- -12V @ 1A
- 1 Variable: 0 - 20V @2A
- (NEW XP581A version goes down to 0)
- Fully Regulated and Short Protected
- Fully Regulated and Short Protected
- Voltage and current meters
- Ideal for laboratory; service shops and hobbyists
- Low cost and high value

click any image to enlarge

Flow Control Valv

ES-B6800

For Commercial and Industrial Applications

Job Name _____ Contractor _____

Job Location _____ Approval _____

Engineer _____ Contractor's P.O. No. _____

Approval _____ Representative _____

Series B6800, B6801

3-Piece, Full Port, Brass Ball Valves

Sizes: ¼" – 2" (8 – 50mm)

Series B6800, B6801 3-Piece, Full Port, Brass Ball Valves feature an in-line maintenance design that offers serviceability of all operating parts without disturbing the rigid pipeline system. The B6800, B6801's full port orifice ensures maximum flow capacity, while Durafill® seats, chrome plated brass ball and blow-out proof stem provide maximum safety and highest operating pressure and temperature limits.

Features

- 3-piece, lift-out design
- Carbon/glass reinforced PTFE Durafill® valve seats
- Chrome plated brass ball
- Blow-out proof, pressure retaining stem
- Standard actuator mounting pads
- High cycle life reinforced PTFE stem packing seal and thrust washer
- Vinyl insulator on heavy duty, zinc plated carbon steel handles
- Low operating torque
- Adjustable stem packing gland
- Each valve factory tested

Models

B6800 ¼" – 2" (8 – 50mm) threaded NPT end connections

B6801 ½" – 2" (15 – 50mm) solder end connections*

Specifications

A 3-piece full port brass ball valve to be installed as indicated on the plans. The valve must have a blowout proof stem, reinforced Durafill seats, reinforced PTFE stem packing, and chrome plated brass ball. Pressure rating no less than 600psi (41 bar) WOG non-shock, 150psi (10 bar) WSP for ¼" – 1" and 400psi (28 bar) WOG non-shock, 125psi (8.6 bar) WSP for 1¼" – 2". Valve must conform to MSS-SP-110 and shall be a Watts Series B6800 (threaded) or B6801 (solder).

*This valve is designed to be soft soldered into lines without disassembly, using a low temperature solder (420°F/216°C). Other solders such as 95/5 tin antimony (460°F/238°C) can be used. However, extreme caution must be used to prevent seat damage. Higher temperature solders will damage the seat material. ANSI B.16.18 states that the maximum operating pressure of 50-50 solder connections is 200psi (14 bar) at 100°F (38°C) and decreases with higher temperatures.
Apply heat with the flame directed **AWAY** from the center of the valve body. Excessive heat can harm the seats. After soldering, the packing nut may have to be tightened.

BAA/ARRA Compliant**

**This product complies with the Buy American Act and The American Recovery and Reinvestment Act. For more information, visit watts.com.

Options

Suffix	
Z15 –	Less lever and nut
XH –	Extended handle
G –	Grounded ball
GS –	Grounded ball and stem
SS –	316 Stainless steel ball and stem
OV –	Oval handle
RH –	Round handle
SH –	Stainless steel handle and nut
SE –	Safety exhaust (unidirectional), see literature ES-B6800SE
(01) VT –	Virgin PTFE seat and seal
BS –	Balancing handle stops
LL –	Latch-Lok handle (304 SS)
TH –	Tee handle
LC –	Latch-Lok handles latch and lock in "closed" position only


Exclusive Latch-Lok Handle (option LL)

Pressure – Temperature

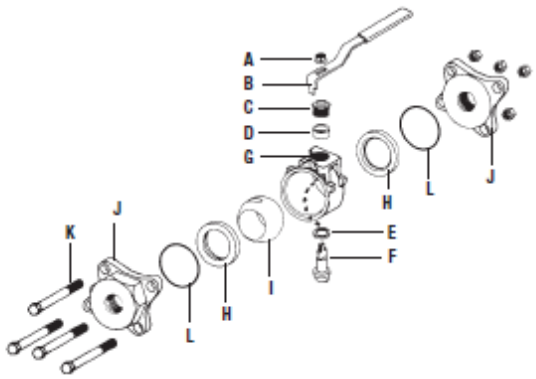
Temperature Range: 0°F – 450°F (-18°C – 232°C)

¼" – 1" (8 – 25mm)
600psi (41 bar) WOG non-shock
150psi (10 bar) WSP

1¼" – 2" (32 – 50mm)
400psi (28 bar) WOG non-shock
125psi (8.6 bar) WSP

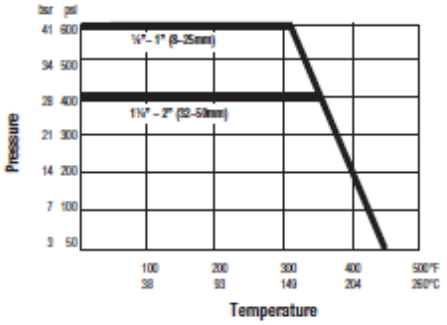
WATTS®

Materials

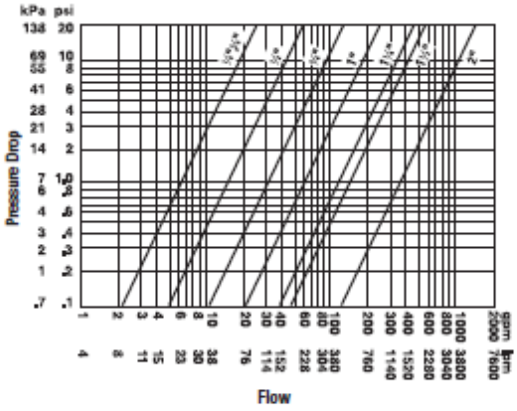


A	**Handle Nut**	Zinc Plated Carbon Steel
B	**Handle**	Zinc Plated Carbon Steel with Vinyl Insulator
C	**Packing Nut**	Brass ASTM B16, C36000
D	**Stem Packing**	Glass Reinforced PTFE
E	**Thrust Bearing**	Glass Reinforced PTFE
F	**Stem**	Brass ASTM B16, C36000
G	**Body**	Forged Brass ASTM B124
H	**Seats**	Carbon/Glass Reinforced PTFE Durafill®
I	**Ball**	Chrome Plated Brass
J	**Adapter**	Forged Brass ASTM B124
K	**Body Bolts & Nuts**	Zinc Plated Carbon Steel
L	**Body Seals**	PTFE

Valve Seat Rating



Pressure Drop vs. Flow



Dimensions — Weights

B6800

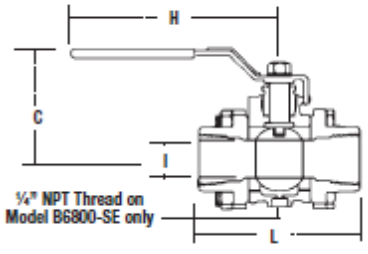
SIZE (DN)		C Center to Handle		H Radius of Handle		I Ball Orifice		L End to End		WEIGHT	
in.	mm	in.	mm	in.	mm	in.	mm	in.	mm	Lbs.	Kg.
¼	8	1¾	44	3⅞	98	⅜	10	2⅜	60	1.1	.5
⅜	10	1¾	44	3⅞	98	⅜	10	2⅜	60	1.1	.5
½	15	1¾	44	3⅞	98	½	13	2⅜	60	1.1	.5
¾	20	2¼	57	4½	114	¾	19	3¼	83	2.5	1.1
1	25	2¾	70	6⅛	156	1	25	3⅞	98	4.1	1.9
1¼	32	3	76	6⅛	156	1¼	32	4½	114	6.3	2.9
1½	40	3½	89	8	203	1½	38	5	127	9.3	4.2
2	50	3⅞	98	8	203	2	51	6⅝	168	13.8	6.3

**B6801*

½	15	1¾	44	3⅞	98	½	13	2⅜	60	1.1	.5
¾	20	2¼	57	4½	114	¾	19	3¼	83	2.5	1.1
1	25	2¾	70	6⅛	156	1	25	3⅞	98	4.1	1.9
1¼	32	3	76	6⅛	156	1¼	32	4½	114	6.3	2.9
1½	40	3½	89	8	203	1½	38	5	127	9.3	4.2
2	50	3⅞	98	8	203	2	51	6⅝	168	13.8	6.3

*See solder instructions on front.

SIZE (DN)		TORQUE		
in.	mm	in.-Lbs.	N-m	Cv
¼-⅜	8-10	60	6.8	6
½	15	60	6.8	15
¾	20	150	16.9	30
1	25	200	22.6	60
1¼	32	250	28.2	110
1½	40	320	36.2	130
2	50	500	56.5	360



WATTS®

A Watts Water Technologies Company

ISO 9001-2008 CERTIFIED

USA: No. Andover, MA • Tel. (978) 688-1811 • Fax: (978) 794-1848 • www.watts.com
Canada: Burlington, ONT. • Tel. (905) 332-4090 • Fax: (905) 332-7068 • www.wattscanada.ca

NI 9219:

Technical Sales
United States
(866) 531-6285
info@ni.com

NI 9219

24-Bit Universal Analog Input

- 250 Vrms channel-to-channel isolation
- Built-in quarter, half, and full-bridge support
- Built-in voltage and current excitation
- Thermocouple, RTD, resistance, voltage, and current measurements
- CJC per channel for accurate thermocouple measurement
- 100 S/s/ch simultaneous inputs (50S/s/ch for Thermocouple)

Overview

The National Instruments 9219 is a 4-channel universal C Series module designed for multipurpose testing in any NI CompactDAQ or CompactRIO chassis. With the NI 9219, you can measure several signals from sensors such as strain gages, RTDs, thermocouples, load cells, and other powered sensors. The channels are individually selectable, so you can perform a different measurement type on each of the four channels. Measurement ranges differ for each type of measurement and include up to ±60 V for voltage and ±25 mA for current. Please see the manual for detailed specifications and ranges.

Because of the driver design, the NI 9219 does not limit the overall speed of an NI CompactDAQ system when used with faster sampling modules.

With 250 Vrms of channel-to-channel isolation, the NI 9219 protects not only the surrounding modules, chassis, and connected computer system but also the other channels within the same module. In addition to increased safety, channel-to-channel isolation eliminates problems associated with ground loops.

The NI 9219 uses 6-position spring terminal connectors in each channel for direct signal connectivity. You can purchase additional connectors to reduce signal connection time for multiple test units. In addition to extra connectors, a strain relief kit is available to secure the signal wires.

Strain relief backshells for signal wire security and high-voltage protection (qty 4): NI 9972

Extra connectors for 6-position connector modules (qty 10): NI 9973

Specifications

Specifications Documents

- Specifications (2)
- Data Sheet

Specifications Summary

General

Product Name	NI 9219
Product Family	Industrial I/O
Form Factor	CompactDAQ , CompactRIO
Part Number	779781-01
Operating System/Target	Real-Time , Windows
Measurement Type	Resistance , Temperature , RTD , Bridge-based sensor , Thermocouple , Voltage , Current
Isolation Type	Ch-Ch Isolation
RoHS Compliant	Yes
Signal Conditioning	Current excitation , Cold-junction compensation , Bridge completion , Voltage excitation , 0-20 mA current input
Analog Input	
Channels	4 , 0
Single-Ended Channels	0
Differential Channels	4
Resolution	24 bits
Sample Rate	100 S/s
Max Voltage	60 V
Maximum Voltage Range	-60 V , 60 V
Maximum Voltage Range Accuracy	243 mV
Minimum Voltage Range	-0.125 V , 0.125 V
Minimum Voltage Range Accuracy	271 µV
Maximum Current Range	-0.025 A , 0.025 A
Maximum Current Range Accuracy	152 µA
Simultaneous Sampling	Yes
Excitation Voltage	2.5 V
Bridge Configurations	Half Bridge , Full Bridge , Quarter Bridge
Analog Output	
Channels	0
Digital I/O	
Bidirectional Channels	0
Input-Only Channels	0

NI 9205:

Technical Sales
United States
(866) 531-6285
info@ni.com

NI 9205

32-Ch ±200 mV to ±10 V, 16-Bit, 250 kS/s Analog Input Module

- 32 single-ended or 16 differential analog inputs
- 16-bit resolution; 250 kS/s aggregate sampling rate
- ±200 mV, ±1, ±5, and ±10 V programmable input ranges
- Hot-swappable operation; overvoltage protection; isolation; NIST-traceable calibration
- -40 to 70 °C operating range
- Spring terminal or D-Sub connectivity

Overview

The National Instruments 9205 is a C Series module, for use with NI CompactDAQ and CompactRIO chassis. The NI 9205 features 32 single-ended or 16 differential analog inputs, 16-bit resolution, and a maximum sampling rate of 250 kS/s. Each channel has programmable input ranges of ±200 mV, ±1, ±5, and ±10 V. To protect against signal transients, the NI 9205 includes up to 60 V of overvoltage protection between input channels and common (COM). In addition, the NI 9205 also includes a channel-to-earth-ground double isolation barrier for safety, noise immunity, and high common-mode voltage range. It is rated for 1,000 Vrms transient overvoltage protection.

There are two connector options for the NI 9205; a 36-position spring terminal connector for direct connectivity or a 37-position D-Sub connector. To add strain relief and high-voltage protection to the 36-position terminal of the NI 9205, NI recommends the NI 9940 strain-relief connector accessory.

The NI 9205 with D-Sub option has an industry-standard 37-position D-Sub connector that provides a low-cost cabling option to a wide variety of accessories available from NI or other vendors. A number of vendors who provide custom D-Sub cable fabrication services, can provide cables with a pin-out that matches your exact application needs. The NI 9933 (or other 37-pin D-Sub connector) is required for use with the NI 9205 with D-Sub. The NI 9933 includes a screw-terminal connector with strain relief as well as a D-Sub solder cup backshell for creating custom cable assemblies.

Specifications

Specifications Documents

- Specifications
- Data Sheet

Specifications Summary

General	
Product Name	NI 9205
Product Family	Industrial I/O

Form Factor	CompactDAQ , CompactRIO
Part Number	779519-01
Operating System/Target	Real-Time , Windows
Measurement Type	Voltage
Isolation Type	Ch-Earth Ground Isolation
RoHS Compliant	No
Analog Input	
Channels	16 , 32
Single-Ended Channels	32
Differential Channels	16
Resolution	16 bits
Sample Rate	250 kS/s
Max Voltage	10 V
Maximum Voltage Range	-10 V , 10 V
Maximum Voltage Range Accuracy	6220 µV
Minimum Voltage Range	-0.2 V , 0.2 V
Minimum Voltage Range Accuracy	157 µV
Simultaneous Sampling	No
Analog Output	
Channels	0
Digital I/O	
Bidirectional Channels	0
Input-Only Channels	0
Output-Only Channels	0
Number of Channels	0
Counter/Timers	
Counters	0
Physical Specifications	
Length	9 cm
Width	2.3 cm
I/O Connector	36-position spring terminal , 37-pin D-Sub

APPENDIX H: DESIGN SUMMARY OF A POSSIBLE HYBRID MOTOR

Table of Contents

Variable Definitions 139

Introduction 139

Propellants 140

Nitrous Oxide 140

Equation 1: Vapor pressure 141

Equation 2: Density of the saturated liquid 141

Equation 3: Density of the saturated gas 141

Equation 4: Specific enthalpy of the saturated liquid 141

Equation 5: Specific enthalpy of the saturated gas 141

Equation 6: Constant pressure specific heat of saturated liquid 141

Equation 7: Constant pressure specific heat of the saturated gas 141

Table 1: Constants used in above equations 142

Paraffin 142

Equation 8: Fuel regression rate, where Gox is the oxidizer flux in g/cm^2*s 143

Stoichiometric Oxidizer to Fuel Ratios 143

Figure 1: Gamma as a function of oxidizer to fuel ratio, generated by CP Technology's CHEM program 145

Figure 2: Adiabatic flame temperature as a function of oxidizer to fuel ratio, generated by CP Technology's CHEM program 145

Figure 3: Characteristic velocity as a functon of oxidizer to fuel ratio, generated by CP Technology's CHEM program 146

Simulating the Motor Performance 146

Figure 4: Overall simulation flow chart 147

Determining Nozzle Geometry 147

Equation 9: Ideal gas law 148

Equation 10: Isentropic expansion 148

Equation 11: Changes in enthalpy through the isentropic expansion 148

Equation 12: Speed of sound 148

Equation 13: Mach number 148

Equation 14: Gas velocity at the exit cone of the nozzle 148

Equation 15: Expansion ratio 148

Combustion Chamber Modeling 149

Equation 16: Characteristic velocity defined 150

Equation 17: Thrust 150

Equation 18: Specific impulse 150

Equation 19: Characteristic length 150

Oxidizer Tank Modeling 150

Liquid Emptying Regime 151

Equation 20 151

Equation 21 151

Figure 5: Control volume used in tank drainage analysis 151

Equation 22 151

Equation 23 151

Equation 24 151

Equation 25 152

Equation 26 152

Equation 27 152
Equation 28 152
Equation 29 152
Figure 6: Interpolation to find temperature at a discrete time increment .. 153
Gas Emptying Regime 153
Equation 30: Ideal gas law including compressibility 153
Figure 8: Compressibility factor for nitrous oxide 154
Equation 31: The temperature ratio redefined 155
Equation 32 155
Equation 33 155
Figure 9: Comparison of nitrous oxide drainage simulations to real data. 156
Mass Flow between Oxidizer Tank and Combustion Chamber Modeling 156
Thrust Control 156
Figure 10: Control system block diagram 157
Equation 34: Mass flow from pressure and temperature 157
Equation 35: Equation 14 re-written 158
Equation 36: Equation 34 re-written 158
Equation 37: Simplified thrust equation 158
Simulation Results 159
Table 2: Constants and Initial Values used in the Simulations 159
Figure 11: Motor simulation with no controller 160
Figure 12: Motor simulation data with a controller 162
Hybrid Motor Physical Design 164
Figure 13: Rendering and description of possible motor design 164
Oxidizer Tank 165

Combustion Chamber 165

Combustion Chamber Casing..... 165

Figure 14: Rendering of combustion chamber nozzle end..... 165

Nozzle and Post Combustion Chamber 166

Figure 15: Rendering of nozzle and post combustion chamber 166

Combustion Chamber Upper Bulkhead 167

Figure 16: Combustion chamber injector bulkhead..... 167

Variable Definitions

V	Velocity
v	Specific Volume
ρ	Density
T	Temperature
P	Pressure
Z	Compressibility Factor
U	Internal Energy
u	Specific Internal Energy
h	Specific Enthalpy
k	Ratio of Specific Heats
M	Molecular Mass
m	Mass
1	Generally means initial value or value in combustion chamber
2	Generally means final value or value at nozzle exit cone
t	Time, or subscripted as value in nozzle throat
A	Area
R	Ideal Gas Constant
ε	Nozzle Expansion Ratio
C*	Characteristic Velocity
L*	Characteristic Length
a	Speed of Sound
L	Length, or VF Multiplier

Introduction

This report gives an overview of the proposed Q4200 hybrid rocket motor for team Ursa's Delta P sounding rocket as the design currently stands. It also reviews how these designs were developed. The honors thesis scope is the beginning of the work to validate this design, and including this appendix that details the design of an actual motor gives the rest of the thesis a better sense of relevance and perspective.

The general design scheme of this particular motor is to use self-pressurizing nitrous oxide fed into a combustion chamber containing paraffin wax. All simulation of motor performance has been done from scratch in MATLAB, with the exception of C* and chamber temperature calculations. These have been computed using John Wickman's CHEM program. A combustion efficiency of 90% has been assumed. Details regarding fuel regression rate, efficiencies, L* computations, and similar combustion parameters are all detailed in this report. What may be the most difficult part of the motor design is the implementation of the thrust control system which is discussed in the thesis.

Physically, the motor is constructed of extruded aluminum 6061-T6 tubing. The nitrous oxide tank would be used as the airframe of the rocket booster section. Rough calculations show that the current oxidizer tank has ample factor of safety to be used as a

structural component. The combustion chamber is a separate pressure vessel from the oxidizer tank. The tank end closers are flat plates of aluminum with double O-ring seals. Detailed finite element analysis has been completed on these pieces. The bulkheads are held within the tubing via spiral snap rings. The nozzle will be canvas phenolic with a graphite throat insert.

Propellants

The choice of propellants, nitrous oxide and paraffin should be justifiable. They are becoming a more and more common combination of propellants, for several reasons. Mainly, because nitrous oxide is self-pressurizing and paraffin has an extremely high regression rate. Similar regression rates can't be achieved with HTPB, especially without lots of experience and plenty of additives.

Nitrous Oxide

Nitrous oxide was the choice oxidizer simply because handling a cryogenic fluid such as liquid oxygen is beyond our ability at this time. In the future liquid oxygen may become a high specific impulse option, but with short development times it is not practical. In addition, handling liquid oxygen on the launch site is impractical for our budget. Curve fit data for many thermodynamic properties are shown in *"Thermo-physical Properties of Nitrous Oxide"* published by IHS. Secondly, nitrous oxide is self-pressurizing. The original plan was to have onboard pressure tank that would maintain 1200psi in the main oxidizer tanks; this did boost the specific impulse up to about 260sec. However, the system added a hefty 40lbs to the total motor weight. The efficiency gain was not enough to outweigh the loss in thrust to weight ratio (in our case). Without the pressure system, max specific impulse dropped to the 220-230sec range. In this scenario the nitrous oxide tank pressure is in the 750-850psi range, depending on the ambient temperature. Exact nitrous pressures are predicted on the in the motor simulation as a function of time. Without the space being taken up by the nitrogen pressure tanks an additional 30% nitrous oxide could also be carried.

A main disadvantage to the nitrous oxide is the high oxidizer to fuel ratio required. This will be discussed later. A main misconception is that the only energy released is from the fuel being oxidized by the oxygen content in the nitrous oxide. In fact, a larger proportion of the total energy is from the nitrous oxide decomposing. All of this was taken into account by CP technology's CHEM program. Once this extra energy is taken into account, the performance of nitrous oxide is not all that bad.

To get thermodynamic properties of the saturated nitrous oxide, curve fits derived in the paper *"Thermo-physical Properties of Nitrous Oxide"* published by HIS were used. The equations are given below, and the referenced constants are in table 1.

$$\log_e\left(\frac{p}{p_c}\right) = \frac{1}{T_r}[b_1(1-T_r) + b_2(1-T_r)^{3/2} + b_3(1-T_r)^{5/2} + b_4(1-T_r)^5].$$

Equation 1: Vapor pressure

$$\log_e\left(\frac{\rho(l)}{\rho_c}\right) = b_1(1-T_r)^{1/3} + b_2(1-T_r)^{2/3} + b_3(1-T_r) + b_4(1-T_r)^{4/3}$$

Equation 2: Density of the saturated liquid

$$\log_e\left(\frac{\rho(g)}{\rho_c}\right) = b_1\left(\frac{1}{T_r}-1\right)^{1/3} + b_2\left(\frac{1}{T_r}-1\right)^{2/3} + b_3\left(\frac{1}{T_r}-1\right) + b_4\left(\frac{1}{T_r}-1\right)^{4/3} + b_5\left(\frac{1}{T_r}-1\right)^{5/3}$$

Equation 3: Density of the saturated gas

$$h(l) = b_1 + b_2(1-T_r)^{1/3} + b_3(1-T_r)^{2/3} + b_4(1-T_r) + b_5(1-T_r)^{4/3}$$

Equation 4: Specific enthalpy of the saturated liquid

$$h(g) = b_1 + b_2(1-T_r)^{1/3} + b_3(1-T_r)^{2/3} + b_4(1-T_r) + b_5(1-T_r)^{4/3}$$

Equation 5: Specific enthalpy of the saturated gas

$$c_p(l) = b_1[1 + b_2(1-T_r)^{-1} + b_3(1-T_r) + b_4(1-T_r)^2 + b_5(1-T_r)^3].$$

Equation 6: Constant pressure specific heat of saturated liquid

$$c_p(g) = b_1[1 + b_2(1-T_r)^{-2/3} + b_3(1-T_r)^{-1/3} + b_4(1-T_r)^{1/3} + b_5(1-T_r)^{2/3}]$$

Equation 7: Constant pressure specific heat of the saturated gas

Tr, and Pr represents the reduced temperature and pressure of the saturated nitrous, where the critical pressure and critical temperature are as follows:

Critical Temperature: 309.57 K
Critical Pressure: 7251. kPa
Critical Density: 452. kg/m3

Table 1: Constants used in above equations

Property	b1	b2	b3	b4	b5	Range of Applicability
Vapor Pressure	-6.719	1.360	-1.378	-4.051		-90C-36C
Liquid Density	1.723	-0.840	0.511	-0.104		-90C-36C
Vapor Density	-1.009	-6.288	7.503	-7.905	629	-90C-36C
Liquid Enthalpy	-200	116.043	-917.225	794.779	-589.587	-90C-36C
Vapor Enthalpy	-200	440.055	-459.701	434.081	-485.338	-90C-36C
Liquid Specific Heat	2.500	.023	-3.801	13.095	-14.518	-90C-36C
Vapor Specific Heat	132.632	.052	-.365	-1.202	.536	-90C-36C

Paraffin

Paraffin is a little harder to justify, as it would be just as easy to fabricate a fuel grain from HTPB or HDPE. The appeal was the ability to get a high regression rate with decent specific impulse. The addition of metal to a HTPB and HDPE grain increases their regression rate, never to the level of carefully formulated paraffin, but the specific impulse reduces since metal increases the molecular weight of the gasses. High gas velocity in the nozzle and low gas density is more efficient than high mass density and lower velocity.

If you can get the paraffin to burn efficiently an increase of roughly 180% in regression rate can be realized over HTPB. This is using just raw, unmixed, versions of paraffin and HTPB; no metalizers or other additives. This is according to AIAA 2011-5680, *"ballistic and rheological characterization of paraffin fuels"* by L.Galfetti. Other AIAA papers show similar results, but this is the best source I've found.

Having this high regression rate means that for an equivalent mass flow of fuel you can have a much shorter fuel grain. Since the needed nitrous oxide volume requires a fairly large diameter tank unless an excessively long rocket is to be built; a short squat fuel grain uses the space much more effectively. Another nice thing about paraffin is that the exponent in the regression rate formula (regression rate=a*Gox^n) is nearly .5, even a little bit lower. This means that the mass flow of oxidizer goes up as the fuel port diameter increases, or at least stays the same. For HTPB and HTPE, even with additives, the exponent is in the .6 to .7 range; which means the fuel mass flow takes a nose dive with an increasing fuel port radius. This makes these fuels ill-suited for the short stubby grain configuration. Exact fuel properties used in simulation are shown in Table 2. The regression rate information shown is from several hybrid experiments from Stanford, shown in their paper: *"Design and Development of a 100km Nitrous Oxide/Paraffin Hybrid Rocket Vehicle."* Additional regression rate information is detailed in the paper:

AIAA 2009-5113 *"Effect of a Diaphragm on Performance and Fuel Regression of a Laboratory Scale Hybrid Rocket Motor Using Nitrous Oxide and Paraffin"* by Matthias Grosse.

Of course, it can't be all good news. Mixing of paraffin and nitrous oxide has been difficult, often resulting in low C* efficiencies. This problem is discussed in AIAA 2009-5113 *"Effect of a Diaphragm on Performance and Fuel Regression of a Laboratory Scale Hybrid Rocket Motor Using Nitrous Oxide and Paraffin"* by Matthias Grosse. He reports a C* efficiency of 86.7% with an L* value of .4m, by modification using diaphragms within the combustion chamber he was able to achieve 96.6% efficiency. We do not have the time to develop such a device. Our motor luckily has a much higher L* value. The exact behavior of our hybrids L* curve is discussed in the results section. An optimistic 90% C* efficiency is being used in the simulations. A concern is that a short, high volume combustion chamber that has the same L* as a long narrow one will not result in an equivalent effect on mixing efficiency. The short wide geometry of our chamber remains a major question, even though our L* values are high (upwards of 1.5m). L* is defined as the volume of the combustion chamber divided by the area of the throat.

The regression rate formula that was used is shown below, Equation 8. The source of the values for the coefficient and exponent were taken from AIAA paper: *"Effect of a Diaphragm on Performance and Fuel Regression of a Laboratory Scale Hybrid Rocket Motor Using Nitrous Oxide and Paraffin"* by Matthias Grosse.

$$\frac{dr}{dt} = A * Gox^{N}$$

where

$A = .000472$ m/s

N = .555

Equation 8: Fuel regression rate, where Gox is the oxidizer flux in g/cm^2*s

Stoichiometric Oxidizer to Fuel Ratios

The exact variety of paraffin has not been decided yet. A microcrystalline structure would be most desirable, in order to keep shrinkage to a minimum, however a cheap and accessible paraffinic wax sources has presented itself to our team. The carbon chain length has very little effect on stoichiometric mixture ratio. Therefore, a simulation could be built and optimized without knowing exact paraffin variety. Below is the stoichiometric fuel ratio for a 30 carbon paraffin chain.

$$(1)C_{30}H_{62} + (91)N_2O \rightarrow (30)CO_2 + (31)H_2O + (91)N_2$$

$$M_{Paraffin30} = 422\frac{g}{mol}$$

$$M_{N_2O} = 44\frac{g}{mol}$$

$$O/F_{mass} = \frac{M_{N_2O}}{M_{Paraffin30}} = \frac{91*44}{1*422} = 9.488$$

Now looking at a 40 carbon paraffin chain:

$$(1)C_{40}H_{82} + (121)N_2O \rightarrow (40)CO_2 + (31)H_2O + (121)N_2$$

$$M_{Paraffin40} = 562\frac{g}{mol}$$

$$M_{N_2O} = 44\frac{g}{mol}$$

$$O/F_{mass} = \frac{M_{N_2O}}{M_{Paraffin30}} = \frac{121*44}{1*562} = 9.473$$

As you can see, the difference is practically negligible.

Many properties in the combustion chamber obviously depend on the oxidizer to fuel ratio. Using the CHEM program flame temperature curves, C* curves, and gamma curves as a function of oxidizer to fuel ratio were found. Figure 1 shows gamma as a curves as a function of oxidizer to fuel ratio were established. Figure 1 shows gamma of the combustion gasses, Figure 2 shows the adiabatic flame temperature, and Figure 3 shows the C* curve.

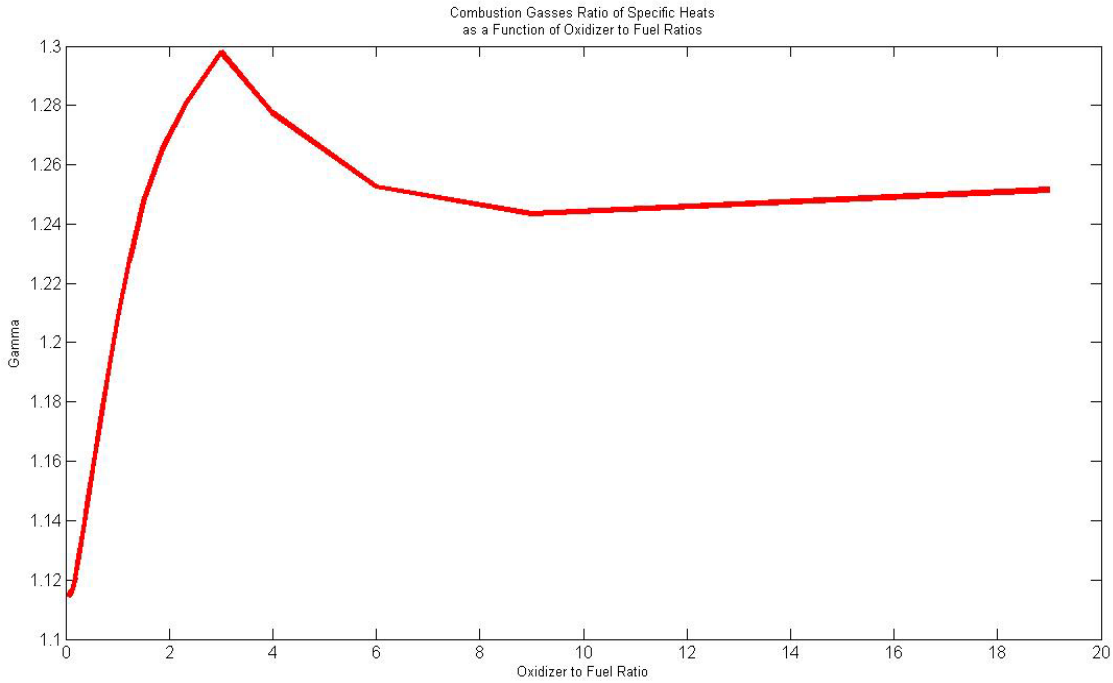


Figure 1: Gamma as a function of oxidizer to fuel ratio, generated by CP Technology's CHEM program

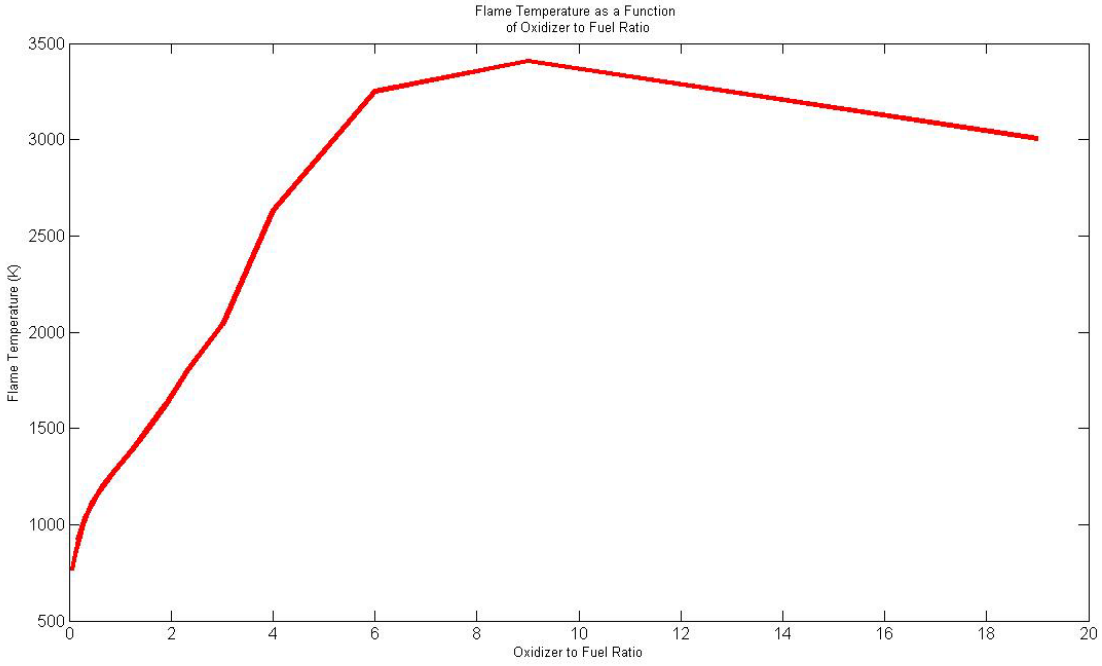


Figure 2: Adiabatic flame temperature as a function of oxidizer to fuel ratio, generated by CP Technology's CHEM program

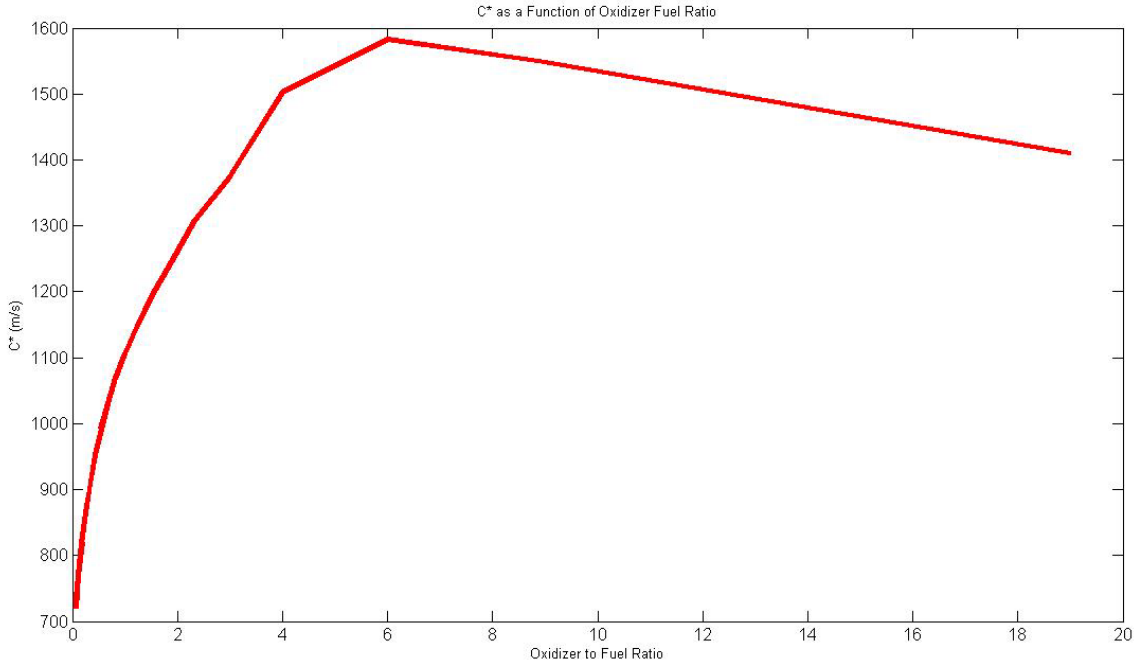


Figure 3: Characteristic velocity as a functon of oxidizer to fuel ratio, generated by CP Technology's CHEM program

Simulating the Motor Performance

Knowing approximately how well the motor will perform before building is absolutely critical. For the purpose of having a full understanding the thermo and physics behind the motor, a scratch built program was used; this was coded into MATLAB. By using our own program made adapting it to specific design subtleties much easier. There are four main components of the simulation, the drainage of the nitrous oxide tank; the combustion chamber behavior; the predicted mass flow of nitrous oxide from the oxidizer tank to the combustion chamber; and the control of the main oxidizer valve position. The valve control system is, of course, the subject of the main report, and will only be lightly touched upon in this report. Just about all the derivations in the following sections are pulled from Sutton's "*Rocket Propulsion Elements.*" Figure 4 shows a very basic block diagram of the simulation process.

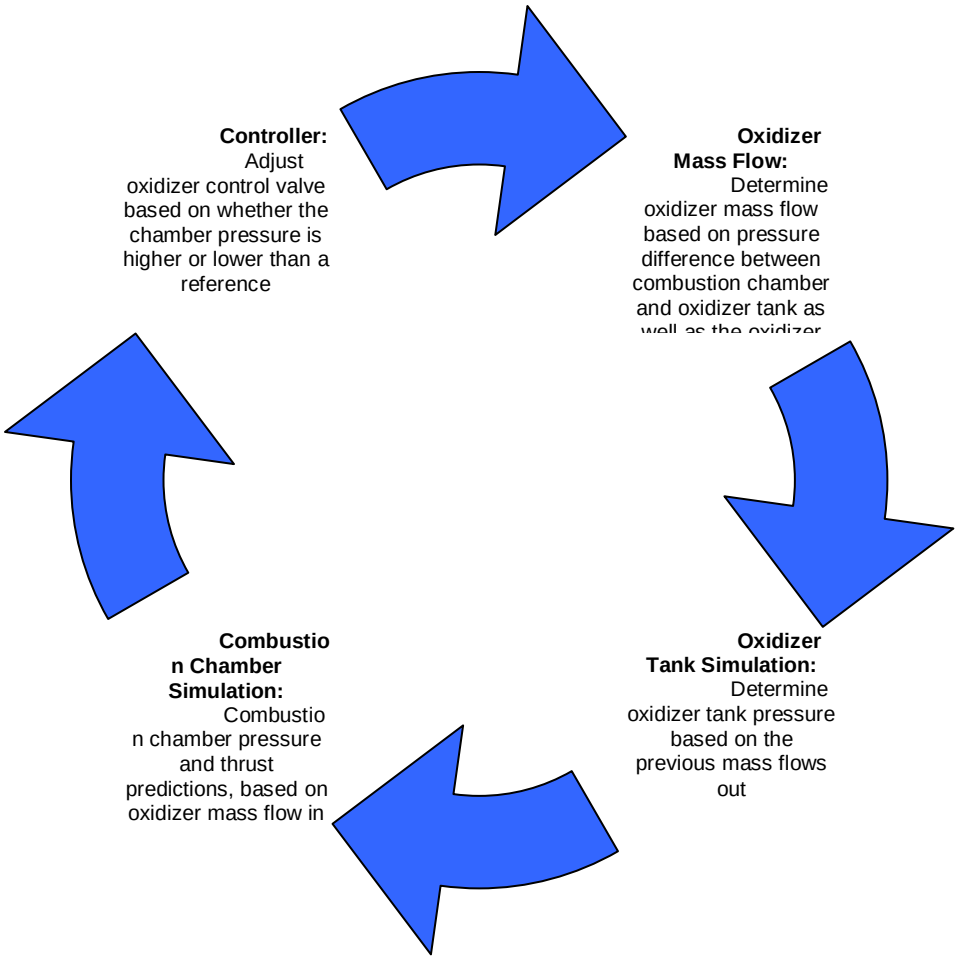


Figure 4: Overall simulation flow chart

Determining Nozzle Geometry

Before either the combustion chamber or oxidizer tank behavior can be modeled accurately we must know the nozzle geometry. This mainly means finding the expansion ratio based on the average chamber pressure and a chosen atmospheric pressure. The average chamber pressure and operating atmospheric pressure are chosen before the simulation is run, and then the expansion ratio can be calculated as a constant to be used in the rest of the code. Throat diameter is guessed, and then the simulation is run to see resulting chamber pressure. If the pressure is too high then the throat diameter is slightly increased and the simulation is run again. Therefore through a short iterative process a desirable throat diameter can be found.

To find the expansion ratio:

$$Pv = RT$$

Equation 9: Ideal gas law

$$\frac{T_x}{T_y} = \left(\frac{P_x}{P_y}\right)^{\frac{k-1}{k}} = \left(\frac{v_y}{v_x}\right)^{k-1}$$

Equation 10: Isentropic expansion

$$h_x - h_y = \frac{1}{2}\left(V_y{}^2 - V_x{}^2\right) = cp\left(T_x - T_y\right)$$

Equation 11: Changes in enthalpy through the isentropic expansion

$$a = \sqrt{kRT}$$

Equation 12: Speed of sound

$$M = \frac{V}{\sqrt{kRT}}$$

Equation 13: Mach number

If Equations 9, 10, and 11 are combined and you define x=1 as the state within the combustion chamber, and y=2 as the state at the exit cone of the nozzle. The velocity in the combustion chamber is considered negligible and is not considered. You get Equation 14, relating velocity at the exit cone to chamber pressure, atmospheric pressure, and chamber temperature.

$$V_2 = \sqrt{\frac{2k}{k-1} RT_1 \left(1 - \left(\frac{P_1}{P_2}\right)^{\frac{k}{k-1}}\right)}$$

Equation 14: Gas velocity at the exit cone of the nozzle

Combining Equation 11 with Equation 14, and 13 you can find the expansion ratio as function of the chamber pressure and atmospheric pressure.

$$\frac{1}{\varepsilon} = \frac{A_t}{A_2} = \left(\frac{k+1}{2}\right)^{\frac{1}{k-1}} \left(\frac{P_2}{P_1}\right)^{\frac{1}{k}} \sqrt{\frac{k+1}{k-1}\left[1 - \left(\frac{P_2}{P_1}\right)^{\frac{k-1}{k}}\right]}$$

Equation 15: Expansion ratio

When a new throat area is chosen before a simulation run the area of the exit cone can be determined, which is needed to get the velocity at that exit plane. The above expansion ratio is only a function of the atmospheric pressure where the nozzle exhausts, and the combustion chamber pressure. Therefore, the nozzle is only optimized at that particular chamber/atmospheric pressure combination; so the average values of those pressures were used. At launch the nozzle will be over expanded, and at full altitude it will be under expanded.

Combustion Chamber Modeling

The modeling of the combustion chamber all starts with a known mass flow of nitrous oxide. This mass flow stays constant for a single small time step before it is recalculated based on a new pressure drop between the nitrous oxide tank and the combustion chamber. The entire motor simulation uses a loop which steps through the simulation duration by small delta times. To explain the modeling of the combustion chamber this section will walk through all the steps in one time loop, these steps start with the above mentioned mass flow of nitrous oxide. At the start of each loop there are several pieces of information that are known:

- Current fuel port bore diameter. For the first time step this is simply the initial bore diameter, for subsequent time steps the diameter is simply assigned as the diameter at the previous time step.
- Oxidizer mass flow. For the first time step this is said to be zero, however, since the initial pressure in the nitrous oxide tank is quite high a new non-zero mass flow is computed at the next time step; and away you go. A new oxidizer mass flow is computed each time step using the difference between combustion chamber pressure and nitrous oxide tank pressure along with the position of the oxidizer control valve.
- Motor geometry. For now, the simplification that motor geometry does not change throughout the burn has been made. In other words the nozzle throat does not erode. See the previous section regarding nozzle geometry calculations.

To make the process easier to visualize the combustion chamber analysis will be broken into a series of steps.

1. Calculate the area of the fuel port using the known port radius at the beginning of the time step
2. Calculate the flux of oxidizer down the fuel port using the area of the port and the known oxidizer mass flow
3. Use the regression rate formula presented in the propellants section of this report to calculate the current regression rate of the fuel grain.
4. Calculate the new fuel port radius using the newly found regression rate; this value will be used in the next time step to calculate the oxidizer mass flux.
5. By knowing the regression rate, the length of the time step and the current fuel port radius a fuel mass flow during the current time step can be computed.
6. Oxidizer to fuel ratio and total mass flow can then be easily found since the oxidizer mass flow is known and the fuel mass flow is known.
7. Using the curve fits shown in Figure 1,2, and 3 along with the newly found oxidizer to fuel ratio a C* value; flame temperature (assumed to be the combustion chamber temperature); and current gamma values for this time iteration can be found.
8. The C* from the curve fit is the value if the motor was burning at 100% efficiency. To correct this, the curved fitted C* is multiplied by the assumed efficiency of 90%.

9. Using Equation 16 and the corrected C* value the chamber pressure can be found.

$$C^* = \frac{P_1 A_t}{\dot{m}}$$

Equation 16: Characteristic velocity defined

10. Using Equation 14 the velocity at the exit cone of the nozzle can be found.
11. Thrust as a function of total mass flow and nozzle exit cone velocity is computed using Equation 17. This neglects pressure thrust.

$$F = \dot{m} V_2$$

Equation 17: Thrust

12. Calculate specific impulse, for interest only, using equation 18.

$$Is = \frac{F}{\dot{m} g}$$

Equation 18: Specific impulse

13. Calculate the characteristic length using equation 19 as a function of total combustion chamber volume and nozzle throat area. V-post is the volume of the combustion chamber beyond the end of the fuel grain.

$$L^* = \frac{A_{port} * L + V_{post}}{A_t}$$

Equation 19: Characteristic length

14. Keep track of total fuel and oxidizer burned as well as the total impulse.

Oxidizer Tank Modeling

Like the combustion chamber, the simulation of the oxidizer tank depends on a constant oxidizer mass flow which is sustained for the duration of a small time step. As mentioned, a new oxidizer mass flow is computed after each time step based on the pressure drop between the oxidizer tank and the combustion chamber as well as the current position of the oxidizer control valve. The important parameters that need to be solved for at each time step are the nitrous oxide tank pressure and nitrous oxide density. There are two regimes of oxidizer tank modeling, one when there is still liquid in the tank, and a second regime once all the liquid has drained from the tank and only gas is flowing into the combustion chamber. I have used a similar method to what is shown in *"Modeling the Nitrous Run Tank Emptying"* by Aspire Space. However, some aspects of their methods seemed a little numerically unstable, so Dr. Boyle from the University of Maine assisted in coming up with a new routine; which is based on the above paper, but very different in some ways. In the MATLAB script there is an "if" statement that

switches the simulation from liquid flow to gas flow once all of the liquid is expelled from the tank.

Liquid Emptying Regime

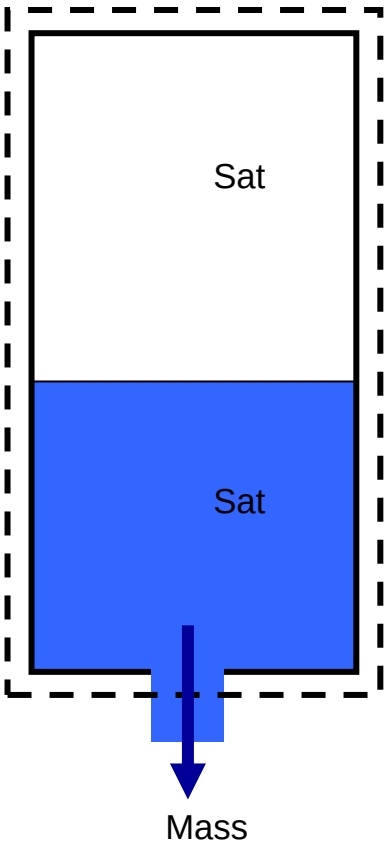


Figure 5: Control volume used in tank drainage analysis

While liquid remains in the tank the nitrous oxide is saturated. There are a couple of really good curve fit regimes for nitrous oxide. Which were presented in Equations 1-7 along with Table 1. There are curves for the vapor pressure, saturated liquid density, saturated vapor density, enthalpy of the saturated liquid, enthalpy of the saturated vapor, and several other properties; all as a function of temperature. The basic strategy to solve for the nitrous oxide pressure and density over time is to take a first law analysis of the entire tank; the control volume used is shown in Figure 5.

Let

$$T_{liq} = T_{vap} = T$$

Equation 20

Write the first law:

$$\dot{W} - \dot{Q} = \frac{\partial}{\partial t}\int u\rho dv + \int h\rho V_{normal}\, dA$$

Equation 21

Work and heat transfer go to zero and the integrals can be shown as:

$$0 = \frac{\partial}{\partial t}U_{C.V.} + \frac{\partial m}{\partial t}h_{exit}$$

Equation 22

Break up the total internal energy within the control volume as such:

$$U_{C.V.} = U_{liq} + U_{vap}$$

Equation 23

$$\frac{\partial U_{C.V.}}{\partial t} = \frac{\partial U_{l}}{\partial t} + \frac{\partial U_{v}}{\partial t}$$

Equation 24

Breaking each of the partials on the RHS of Equation 24 down and then insert into equation 22 results in Equation 25:

$$0 = m_l \frac{\partial u_l}{\partial t} + u_l \frac{\partial m_l}{\partial t} + m_v \frac{\partial u_v}{\partial t} + u_v \frac{\partial m_v}{\partial t} + \frac{\partial m}{\partial t} h_e$$

Equation 25

This equation relies on a differential time step, whereas the simulation uses a finite time step. Equation 17 can be written to use a finite step.

$$0 = m_l' \frac{u_l'' - u_l'}{\Delta t} + u_l' \frac{m_l'' - m_l'}{\Delta t} + m_v' \frac{u_v'' - u_v'}{\Delta t} + u_v' \frac{m_v'' - m_v'}{\Delta t} + h_l' \dot{m}_l$$

Equation 26

All variables in equation 26 are known, a double prime indicates the new value, and a single prime indicates the value at the current step. All the single prime values are the double prime values from the last iteration, so those are known. If you take a small temperature step then you can find all the new values for specific internal energy and enthalpy. However, there is no way of knowing how long it took to take that small temperature step, so the new masses of vapor and liquid are not known; since their values depend on time due to the mass flow out of the vessel. In addition, some of the liquid evaporates to maintain vapor pressure. There is an additional piece of information needed, which are Equations 27, 28 and 29.

$$m_l'' = \frac{V_{Total} - v_v'' m_{Total}''}{v_l'' - v_v''}$$

Equation 27

$$m_{Total}'' = m_{Total}' - \dot{m}\Delta t$$

Equation 28

$$m_v'' = m_{Total}'' - m_l''$$

Equation 29

Now the only unknown value after combining Equations 26, 27, 28 and 29 is the time step; which can be algebraically solved for, albeit messy. In fact it's so messy the algebra was performed in MatchCad and then inserted into the MATLAB code. There is one last problem; we now know the time it takes for a finite temperature change. However, the loop works on a fixed time step, not a fixed temperature step. A temperature at a finite time step can be approximated via interpolation, see Figure 6.

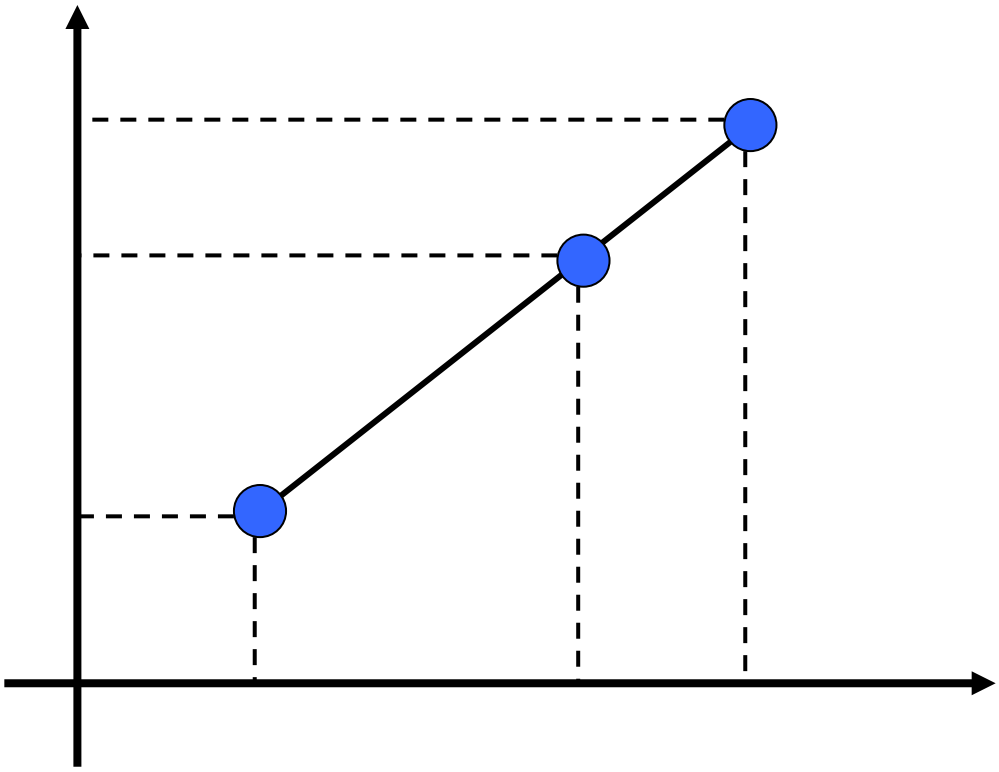


Figure 6: Interpolation to find temperature at a discrete time increment

By repeating the process and using the double prime values as the single prime values during the next time step, the pressure, density, and any other thermodynamic property can be plotted through time.

Gas Emptying Regime

The gas phase of the tank emptying was modeled as an ideal gas that is corrected using a compressibility factor. The compressibility factor diagram is shown in Figure 7. The ideal gas law including a compressibility factor is shown in Equation 30.

$$P = Z\rho RT$$

Equation 30: Ideal gas law including compressibility

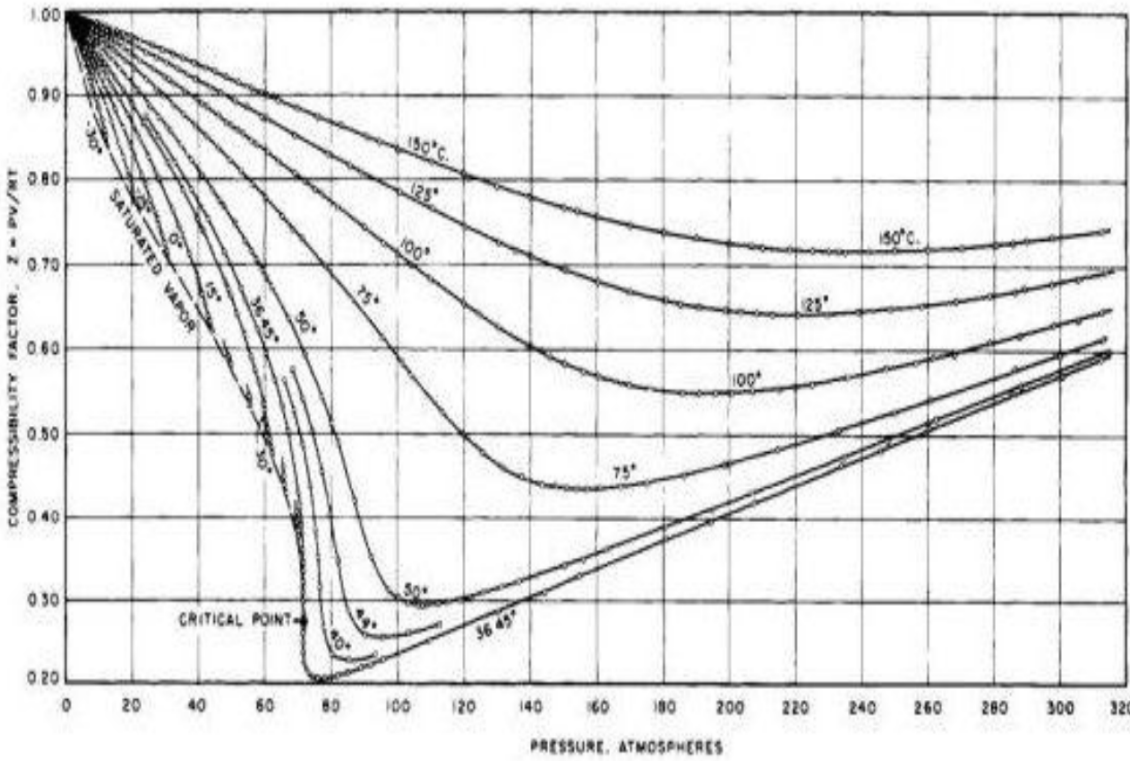


Figure 7: Compressibility factor for nitrous oxide

According to *"Modeling the Nitrous Run Tank Emptying"* by Aspire Space (who have published their simulation results compared to test data) the properties of nitrous oxide follow a line drawn from the critical point to the pressure=0, z=1 point. Since there has been luck using the compressibility factor in the past simulations I figured that it wouldn't hurt to keep using it. The expansion is considered isentropic. The general process of simulation is as follows:

1. The first time the gaseous regime is initiated is when the mass of the liquid nitrous reaches zero.
2. The current and future mass of nitrous in the tank is known by integrating the nitrous oxide mass flow.
3. By looking at equation 10 and substituting Equation 30 for the density ratio on the RHS you can derive Equation 32. Equation 31 is simply found by determining the temperature ratio in terms of Equation 30 and setting density equal to tank volume divided by mass.

$$\frac{T_x}{T_y} = \left(\frac{P_x}{P_y}\right)^{\frac{k-1}{k}} = \left(\frac{v_y}{v_x}\right)^{k-1}$$ **Eq. 10(repeated)**: Isentropic expansion

$$\frac{T_2}{T_1} = \frac{P_2}{P_1}\left(\frac{Z_1 m_1}{Z_2 m_2}\right)$$

Equation 31: The temperature ratio redefined

$$\frac{T_2}{T_1} = \left(\frac{Z_2 m_2}{Z_1 m_1}\right)^{k-1}$$

Equation 32

Equation 32 relates the initial and final temperatures to initial and final compressibility factors as well as to initial and final masses.

4. Equation 32 is all well and good, but the final compressibility factor is unknown because it relies on final pressure, which in turn depends on final temperature. To solve this problem a converging sub-loop be used. To do this we need another equation that relates mass and compressibility factor purely to pressure without a direct dependence on temperature. This is achieved by setting equation 31 and 32 equal to each other.

$$\frac{P_2}{P_1} = \left(\frac{Z_2 m_2}{Z_1 m_2}\right)^{k}$$

Equation 33

5. The sub-loop then behaves as such:
    a. Guess final Z
    b. Find the final temperature using Equation 32, the initial Z is known from the previous time step, the final mass is known from the mass flow and step duration, the initial mass is known from the last time step, and the initial temperature is known from the previous time step. The equation also uses the guessed Z2 value.
    c. Using this new temperature compute the final pressure using Equation 31.
    d. Based on this final pressure find a new final Z using Equation 33.
    e. Find a new final temperature using this newly computed Z value. If this new final temperature is different from the final temperature found with the guessed Z then adjust Z and start again. Repeat until a Z value is converged upon.
6. Once the final Z is found the tank pressure and oxidizer density can be found, which is what's needed from the oxidizer tank simulation.

As explained earlier, the general modeling scheme used was originally taken from the paper: *"Modeling the Nitrous Run Tank Emptying"* by Aspire Space. My numbers seem to match up to theirs fairly well, and they have had good luck matching simulation results to test results; which can be seen in Figure 8 which is data supplied in the above report.

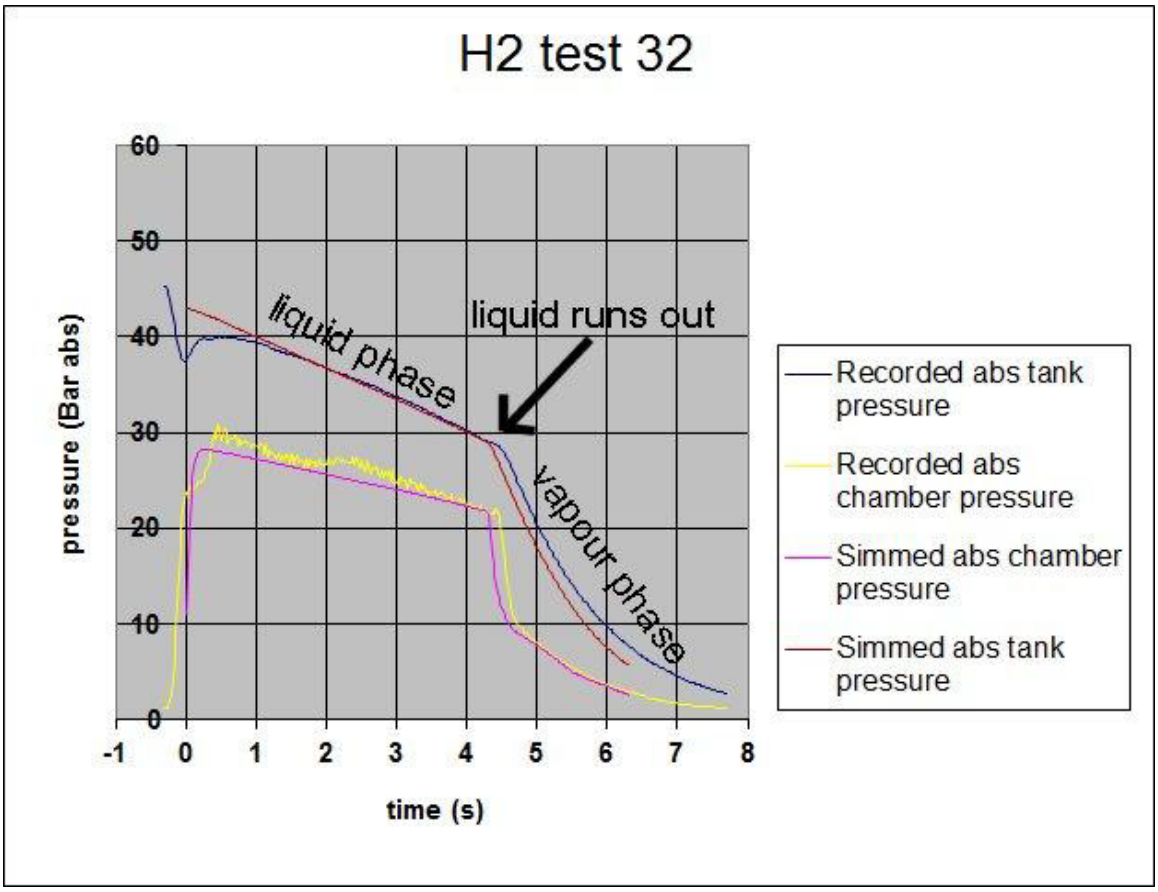


Figure 8: Comparison of nitrous oxide drainage simulations to real data

Mass Flow between Oxidizer Tank and Combustion Chamber Modeling

This step determines the oxidizer mass flow into the combustion chamber based on the pressure drop and oxidizer control valve position. This is covered in more detail within the main report, but is mentioned here for the sake of a complete report. There are several factors in the flow path of the nitrous oxide that prevent the mass flow from being infinite for a given pressure drop.

- Losses in pipe leading to oxidizer control valve
- Losses in the control valve itself, this is a function of valve position
- Losses in pipe leading to the injector
- Losses in the injector

As explained in the main report in more detail, you can use the pressure drop between oxidizer tank and combustion chamber; a valve position; loss coefficients of the tubing and injector; and a VF surface to predict the mass flow of oxidizer through the injector system.

Thrust Control

The last portion of the simulation is the thrust control system, which determines the valve position. This much like the control system used in the main portion of the thesis, but instead of just controlling the mass flow of oxidizer the thrust of the motor is controlled. As can be seen from earlier in this appendix, the thrust is primarily a function of oxidizer mass flow. Therefore controlling the thrust is similar to controlling the mass flow. When in flight the controller will only know the combustion chamber pressure, which is a function of oxidizer mass flow. The next step is to convert the chamber pressure into a thrust measurement. Figure 9 shows the block diagram for the control system.

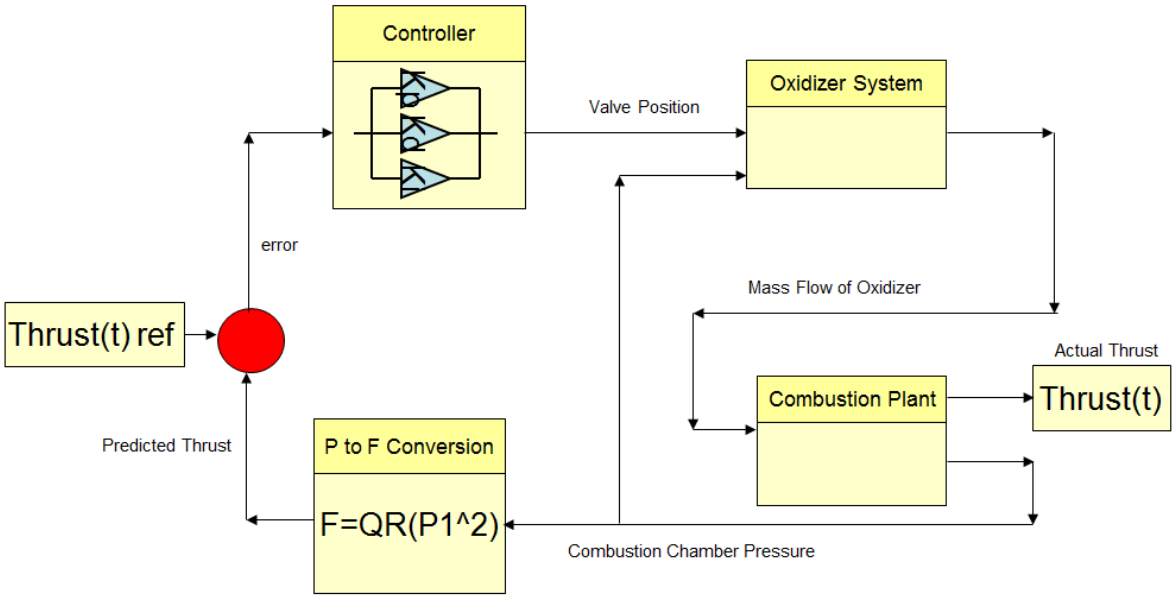


Figure 9: Control system block diagram

One tricky part of the control system is to take the measurable chamber pressure and convert that to the equivalent thrust. When the motor is on the rocket the control system will not be able to measure thrust directly. The problem lies in the fact that thrust depends on several variables besides chamber pressure. Mass flow was known in the simulation, based on regression rate and a few other things, but the controller will not have access to either of these numbers. There is a way of getting the total mass flow indirectly as a function of chamber temperature and pressure. Sutton presents an equation for finding the total mass flow.

$$\dot{m} = \frac{A_t V_t}{v_t} = A_t P_1 k \frac{\sqrt{\left(\frac{2}{k+1}\right)^{\frac{k+1}{k-1}}}}{\sqrt{kRT_1}}$$

Equation 34: Mass flow from pressure and temperature

Equation 34 is equation 3.24 in Sutton's Propulsion Elements.

If we review Equation 14:

$$V_2 = \sqrt{\frac{2k}{k-1} RT_1 \left(1 - \left(\frac{P_1}{P_2}\right)^{\frac{k}{k-1}}\right)}$$ **Eq. 14 (Repeated)**: Speed at the exit cone of the nozzle

Now both components of thrust are known so that we can write the momentum thrust as:

$$F = \dot{m} V_2$$ **Eq. 17 (Repeated)**

By inserting equations 14 and 34 into equation 17 you can find momentum thrust, which is what the controller will compare to a reference thrust. However, you can quickly see that there are chamber temperature terms in the expression as well. Temperature cannot be easily measured, but luckily both functions are more strongly related to P than T. For the sake of the controller we assumed that T was some reasonable average value throughout the operating cycle of the engine. That means all the terms in Equations 14 and 34 are constant except for P. Equation 14 could be now thought of as Equation 36, and Equation 35 could now be thought of as Equation 37.

$$V_2 = \sqrt{D - B * \left(P_1\right)^{\frac{(1-k)}{k}}}$$

Equation 35: Equation 14 re-written

D and B are constants.

$$\dot{m} = Q * \left(P_1\right)$$

Equation 36: Equation 34 re-written

Q is a constant.

If you combine Equations 17, 35, and 36 you get what is shown as the "P to F conversion" in Figure 9. This conversion from pressure to thrust would be handled within the Arduino before the PID controller is applied.

$$F = \sqrt{D - B * \left(P_1\right)^{\frac{(1-k)}{k}}} * Q * P_1$$

Equation 37: Simplified thrust equation

Throughout the static firing process the values of D, B and Q can be fine-tuned to reflect actual motor behavior.

Simulation Results

At this point in time none of the geometry factors have been overly scrutinized, beyond several iterations of hand adjustment to find parameters that seem to work well. Some of the general considerations have been: the need for a large L* to keep mixing good; nozzle geometry that creates an optimal and safe pressure drop; fuel port diameter that keeps the fuel grain from being over-burnt, resulting in a chamber hotspot; and of course, designing the motor to produce about 125,000Ns of impulse and a 2500lbf max thrust. All the results use the regression data presented earlier in the report; and motor geometry shown in table 2.

Table 2: Constants and Initial Values used in the Simulations

Description	Value	Unit
Simulation Duration	20	Sec
Simulation Time Step	.01	Sec
Gravity	9.807	m/s^2
Fuel Grain Length	.4	m
Initial Fuel Port Radius	.05	M
Nozzle Throat Area	.005	m^2
Nozzle Expansion Ratio	4	
Nitrous Tank % Ullage	5	%
C* Efficiency	90	%
Initial Nitrous Oxide Temperature	293	K

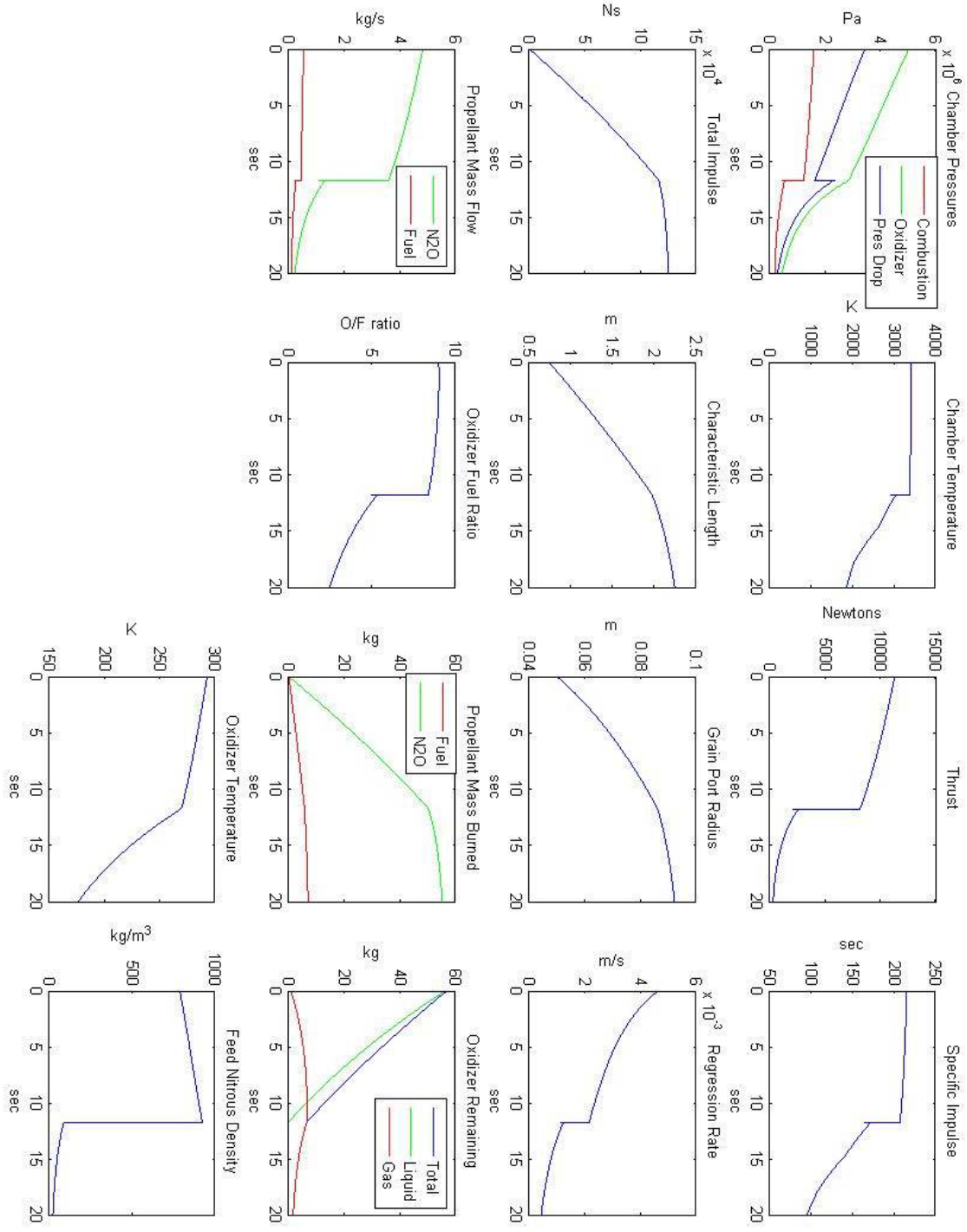


Figure 10: Motor simulation with no controller

Figure 10 contains the plots from a simulation run where the there was no controller used, the valve was set to fully open (90 deg) and left there. This gives a good idea of the motor's natural behavior and also ensures that the pressure differential between the combustion chamber and oxidizer tank won't get too small, resulting in the risk of backflow from the combustion chamber. All the initial values in Table 2 are valid, as well as the regression data earlier in the report. The plots include: chamber pressure, oxidizer pressure, pressure drop between the oxidizer tank and combustion chamber, chamber temperature, thrust, specific impulse, total accumulated impulse, characteristic length, grain port radius, regression rate, nitrous oxide mass flow into the combustion chamber, fuel mass flow off the paraffin fuel grain, oxidizer to fuel ratio, accumulated propellant mass burned, remaining nitrous oxide in run tank, oxidizer tank temperature, and feed nitrous density.

Figure 11 contains the plots from a controlled simulation. All of the same data is presented along with a couple of new plots. One is the valve position rate, which is the speed at which the valve is changing, and the other is the current valve position. The thrust plot also now has a second curve on it which represents a desired thrust profile. This profile is simply a preliminary design which came from the other members of the team. Everything simulates fine, except at the end the motor runs out of nitrous and isn't able to quite match the last portion of the curve. You can also notice that there are saturation points on the control system, one is the max valve position rate, and the others are the max valve positions. Based on max motor rpm the maximum valve position rate was set to 100deg/sec; of course the min and max valve positions are simply open (90 deg) and closed (0 deg). For both tests the initial port radius has been set such that there is approximately a half an inch of remaining paraffin lining the combustion chamber when the firing is complete, this will act as a heat insulator. There is also a phenolic lining around the fuel grain for additional protection.

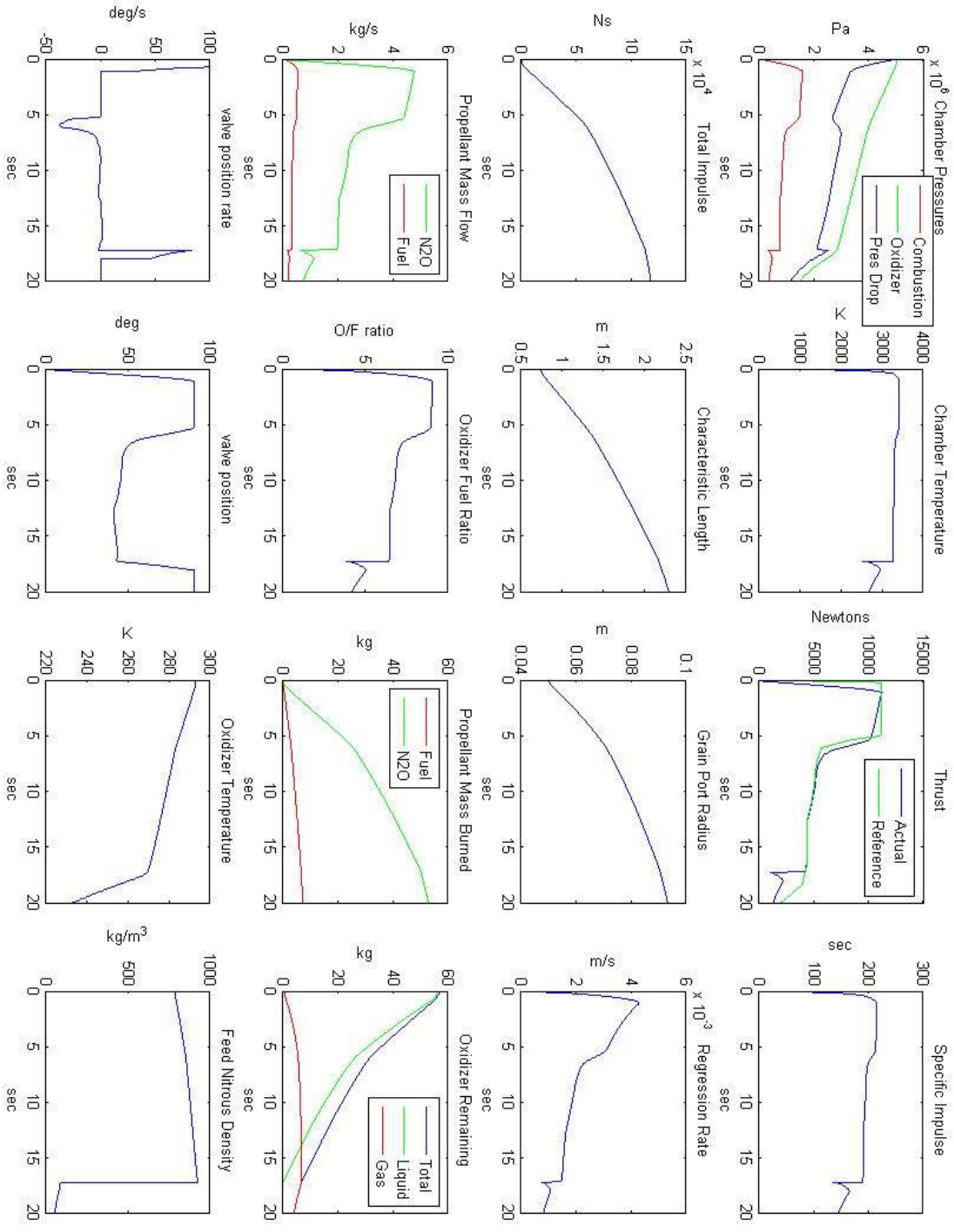


Figure 11: Motor simulation data with a controller

Hybrid Motor Physical Design

Of course MATALB is all well and good, but the physical design is just as important. This document is only concerned with the motor design it does not cover how the motor is mounted within the rocket airframe, or the testing platform; what is worth noting, however, is that no thrust is transmitted through the valve and tubing between the oxidizer tank and combustion chamber. This is simply a proposed design and requires further development. Before such a large motor is manufactured it would also be practical to build several smaller motors and attempt to control them.

As a general overview, the oxidizer tank and the combustion chamber will be formed from aluminum. End closers will be flat aluminum plate held in the tubing with internal snap rings. There will be a small pre and post combustion chamber area in the motor which will be insulated with a thick layer of canvas phenolic. The nozzle will also be canvas phenolic with a small graphite insert in the throat. The fuel grain will have roughly .5in of remaining material at the end of the burn, but this can be adjusted by changing the initial port radius. The fuel grain will also be cast in a thin walled phenolic tube. A concern is the paraffin slumping away from the combustion chamber walls near the end of the burn since the test motor will be horizontal; a possible solution may lay in the design of the fuel grain insulator. If it is constructed in such a way as to give the grain something to latch onto it could support the grain walls near the end of the burn when they are getting thin.

The injector is a radially arranged 12 port, 35deg spray angle, full cone, atomizing injector designed and built by BETE spraying systems. Based on the flow characterizing done by BETE it will be able to handle the required oxidizer mass flow. The oxidizer control valve is simply a Swagelok ½" stainless ball valve, activated by a NEMA 17 step motor which is run through a 30:1 reducing gearbox. The step motor is run by an Omega Engineering 2035 motor controller; which in turn gets the direction, enable, and step commands from an Arduino-Pro microcontroller. This is the part of the design that was tested in the thesis, of course. It is covered in detail in the main report and therefore will not be repeated here.

Figure 11 shows an overall picture of the motor and its components, more detailed sections are to follow. Appendix I displays technical drawings of the entire hybrid motor.

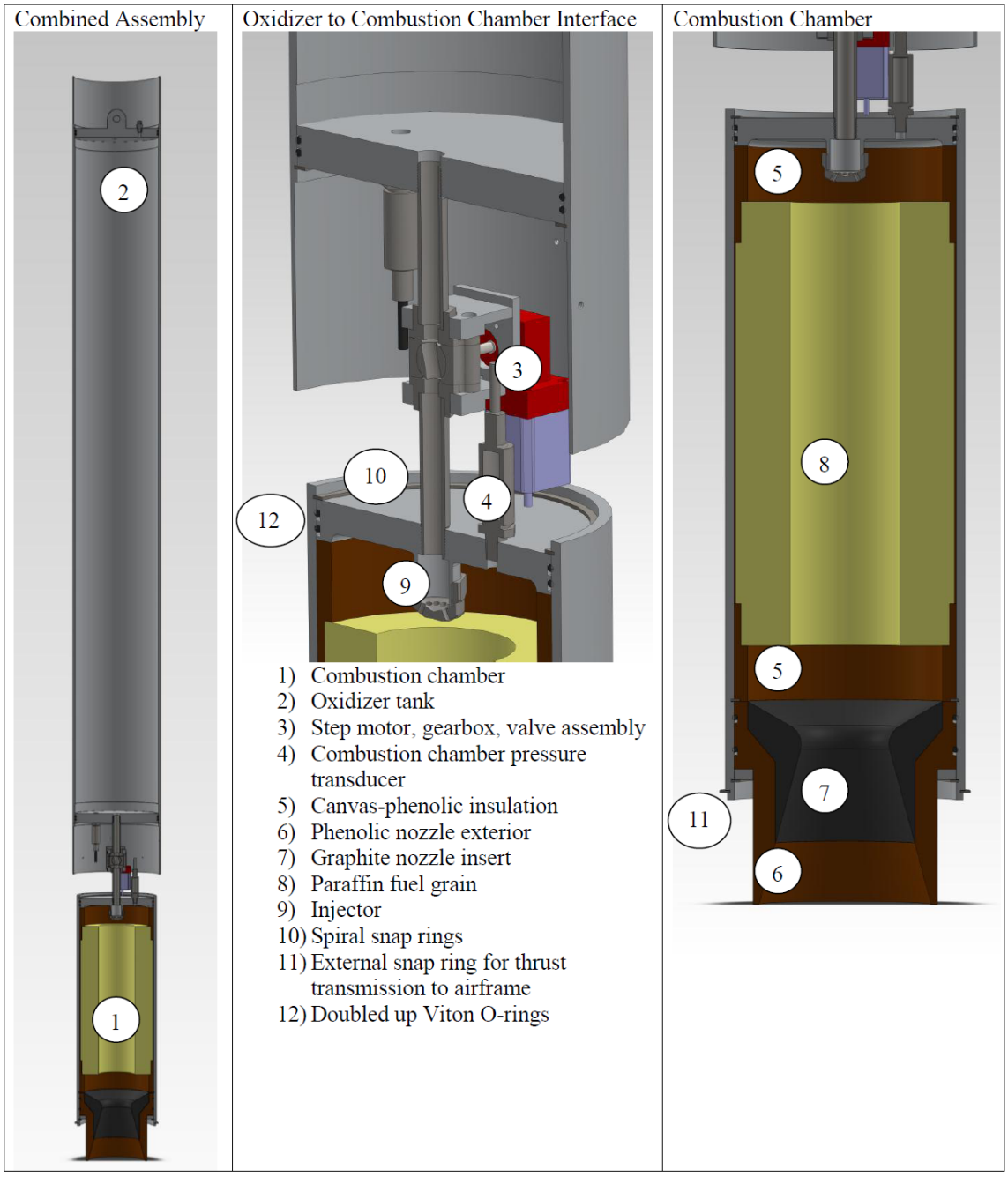


Figure 12: Rendering and description of possible motor design

Oxidizer Tank

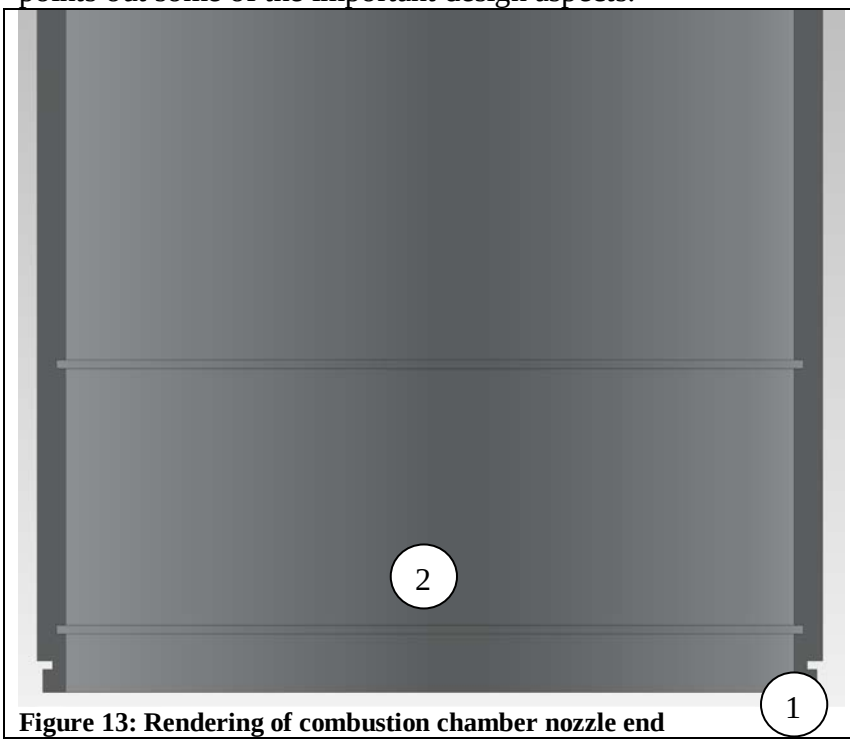
The oxidizer tank design is the same as what is presented in the main thesis, and will not be repeated here.

Combustion Chamber

The combustion chamber will generally see lower pressures than the oxidizer tank; however, it will also see more pressure spikes and unpredictable behavior. Therefore it was also designed to withstand 1000psi, and all factors of safety are based on that pressure. Again, the combustion chamber was designed to fail axially. The radial factor of safety is 3.61, and a minimum factor of safety due to snap ring groove failure is 1.98. Appendix D presents the strength checks performed on the combustion chamber.

Combustion Chamber Casing

Like the oxidizer tank, the combustion chamber casing is formed from extruded 6061-T6 aluminum. For size reasons 8" pipe was selected, which has an OD of 8.625in and a wall thickness of .3125"; the combustion chamber could have been used as the airframe of the rocket as well but it lies in the area where fins would have to be welded to the rocket body. Since welding directly to the combustion chamber would make the temper unknown we decided to avoid this situation by making the combustion chamber fit within the airframe. The thrust from the motor is transmitted to the rocket body by an external snap ring. Figure 13 shows the nozzle end of the combustion chamber and points out some of the important design aspects.

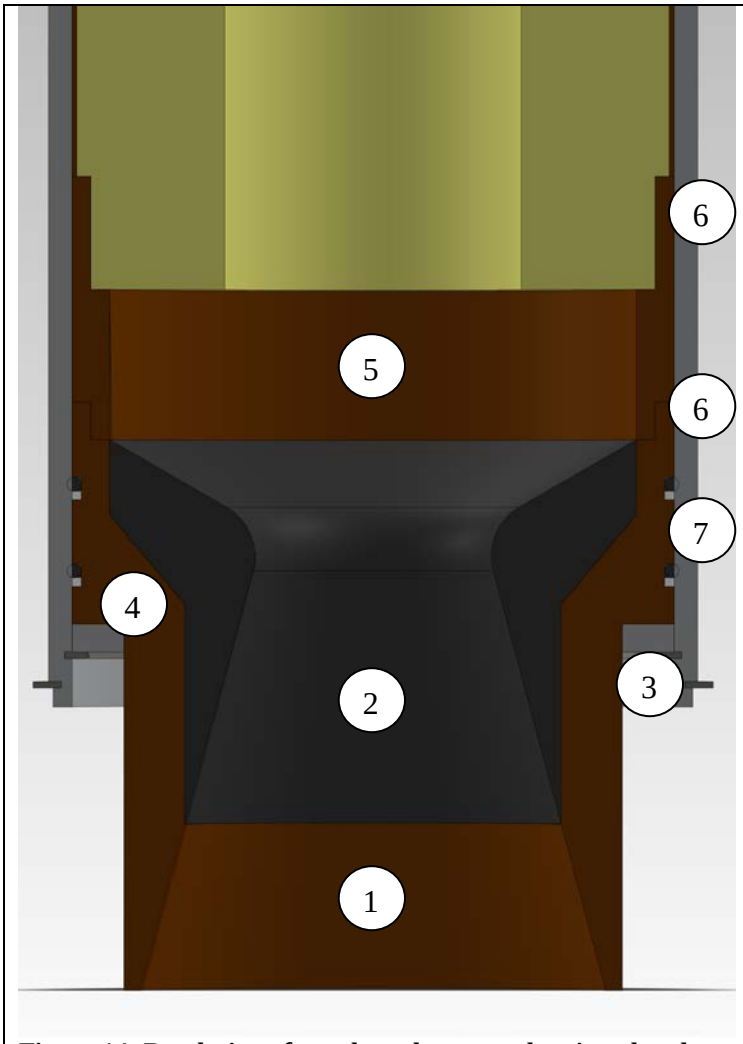


1) Groove for external snap ring
2) Nozzle retaining ring groove

Figure 13: Rendering of combustion chamber nozzle end

Nozzle and Post Combustion Chamber

The nozzle and post combustion chamber is built from a combination of canvas phenolic and graphite. The nozzle throat is machined from graphite and will be bonded to the outer portion of the phenolic nozzle using RTV red high temperature silicon. This construction was used to limit the heat conduction from the nozzle throat to the aluminum combustion chamber. It also helps reduce the temperature gradients in the graphite which could lead to fracture. The other component is the insulator for the post combustion chamber, since something is needed to keep the flame from coming in direct contact with the aluminum combustion chamber. This insulator was simply going to be machined from canvas phenolic, and is made separate from the nozzle so that it can be easily replaced after each burn. Figure 14 displays a rendering of this region of the combustion chamber.



1) Outer nozzle shell (canvas phenolic)
2) Nozzle throat insert (graphite)
3) Nozzle retaining ring
4) Nozzle load distribution ring
5) Post combustion chamber insulator
6) Lap joints to discourage hot gas propagation
7) Viton O-rings

Figure 14: Rendering of nozzle and post combustion chamber

Combustion Chamber Upper Bulkhead

The upper bulkhead of the combustion chamber is very similar to the bulkheads on the oxidizer tank, no FEA results are included with this report but the bulkhead has been shown to be adequate. Figure 15 shows a rendering of the bulkhead, the major difference is the female threaded section where the injector is mounted. Like the oxidizer tank bulkheads the combustion chamber bulkheads are made from 6061 T6 1" thick plate.

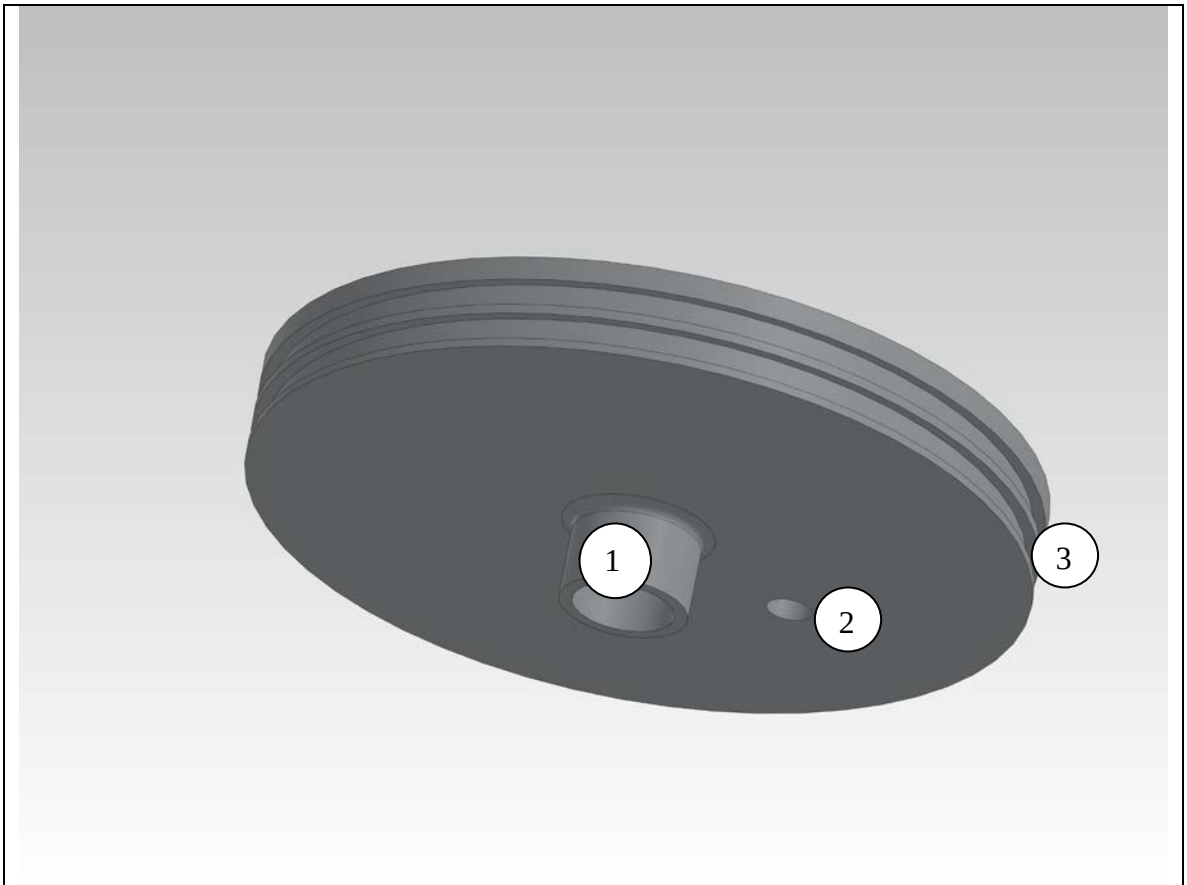
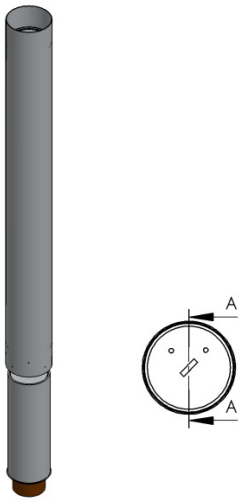


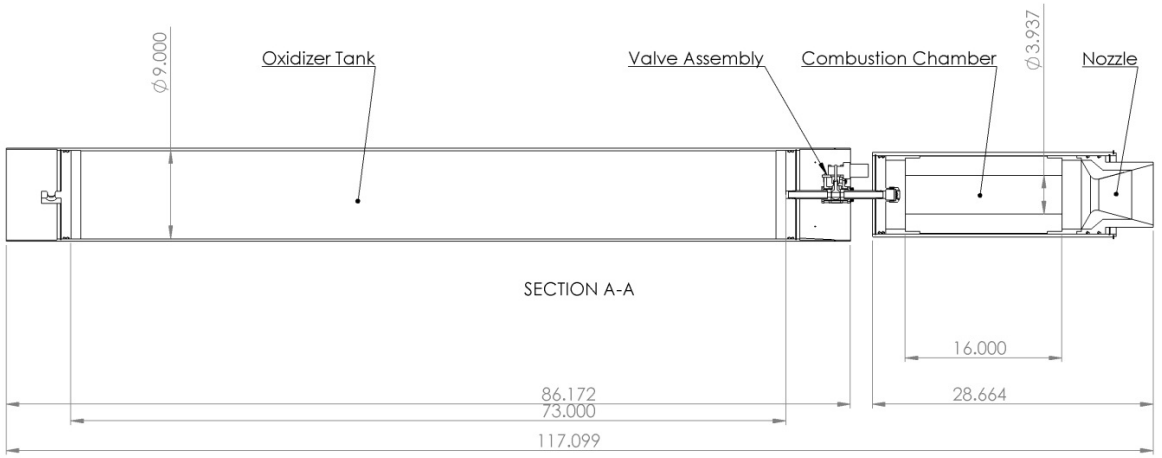
Figure 15: Combustion chamber injector bulkhead

Combustion Chamber Bulkhead, looking at it from the nozzle end of the motor

1) Male 1" NPT connection for the injector
2) Combustion chamber pressure transducer ¼" NPT port
3) O-ring grooves

APPENDIX I: TEST SETUP TECHNICAL DRAWINGS



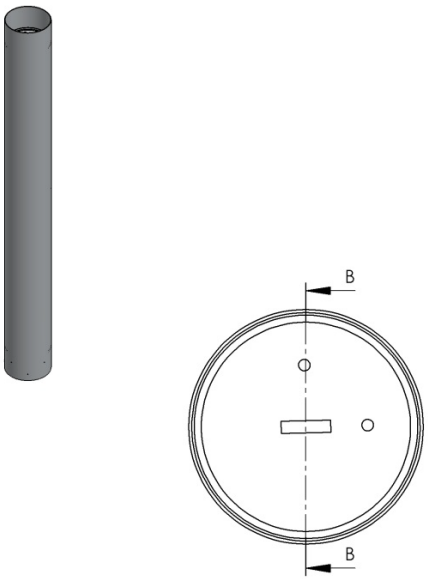


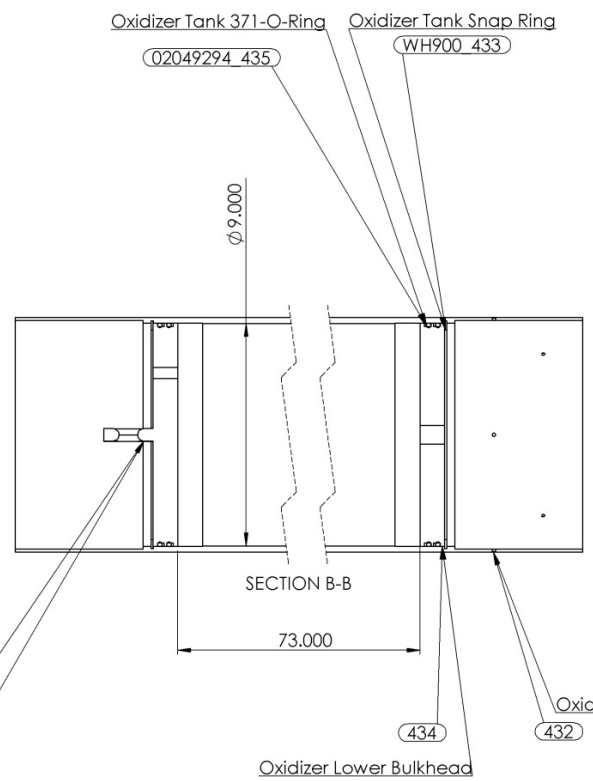
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
	CHECKED		
	ENG APPR.		
	MFG APPR.		
	Q.A.		
MATERIAL	COMMENTS:		
FINISH			
DO NOT SCALE DRAWING			

TITLE: Hybrid Q6250

SIZE	PART NO.	REV
B	450	

SCALE: 1:10 WEIGHT: SHEET 1 OF 32



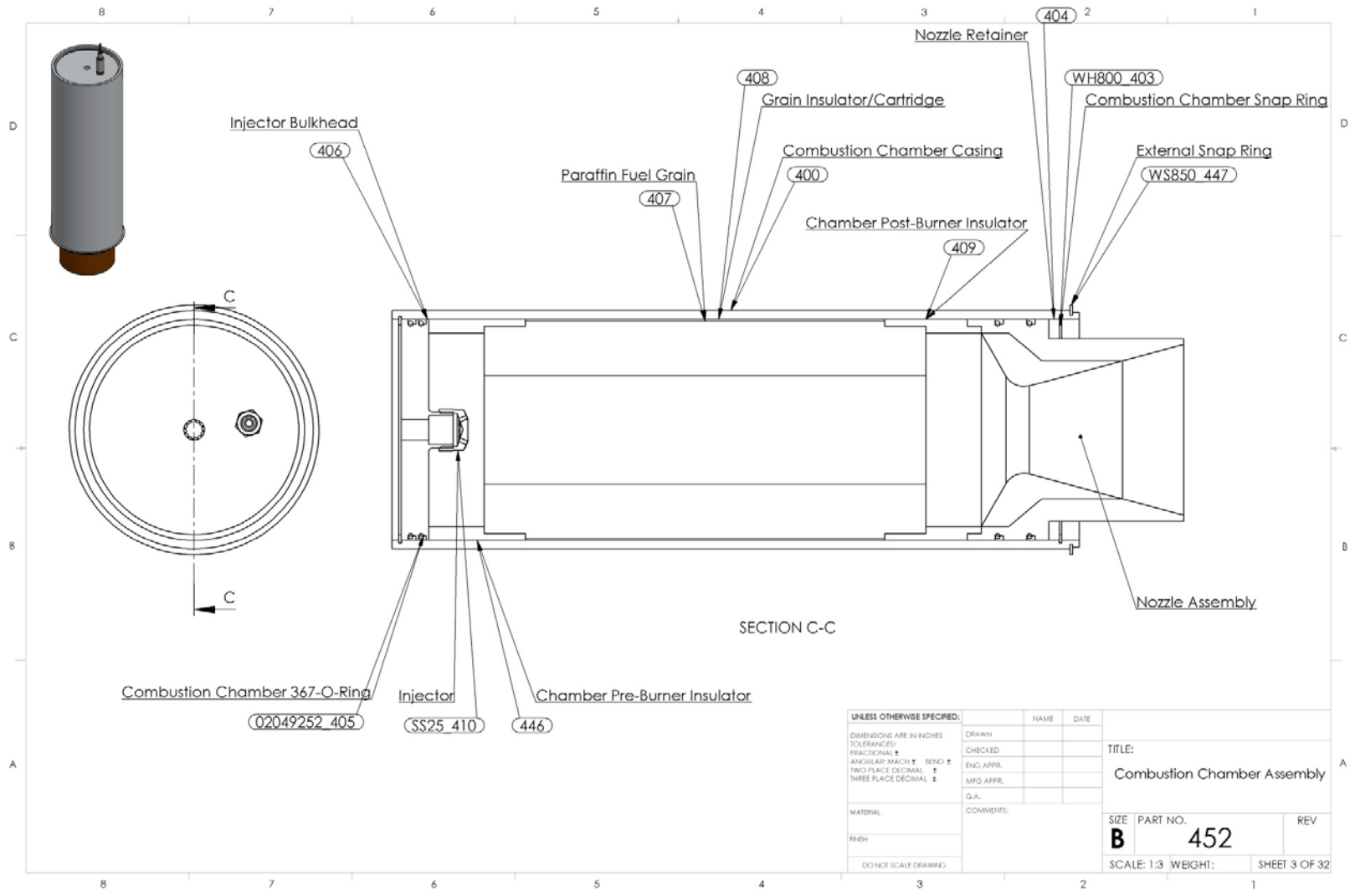


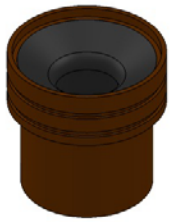
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
	CHECKED		
	ENG APPR.		
	MFG APPR.		
	Q.A.		
MATERIAL	COMMENTS:		
FINISH			
DO NOT SCALE DRAWING			

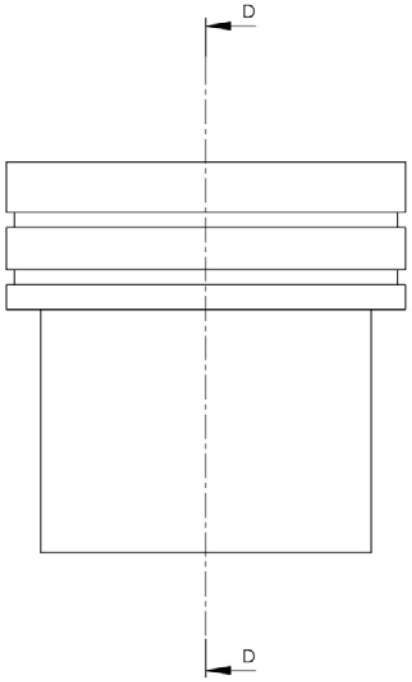
TITLE: Oxidizer Tank Assembly

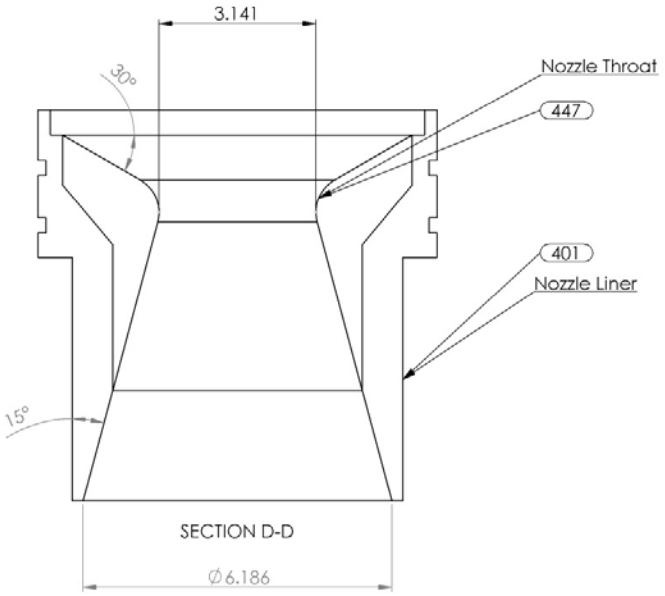
SIZE	PART NO.	REV
B	451	

SCALE: 1:4 WEIGHT: SHEET 2 OF 32


404
Nozzle Retainer
408
Grain Insulator/Cartridge
WH800_403
Combustion Chamber Snap Ring
Injector Bulkhead
406
Combustion Chamber Casing
400
External Snap Ring
WS850_447
Paraffin Fuel Grain
407
Chamber Post-Burner Insulator
409
C
C
Nozzle Assembly
SECTION C-C
Combustion Chamber 367-O-Ring
02049252_405
Injector
SS25_410
Chamber Pre-Burner Insulator
446
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
MATERIAL
FINISH
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
Combustion Chamber Assembly
SIZE
B
PART NO.
452
REV
SCALE: 1:3
WEIGHT:
SHEET 3 OF 32





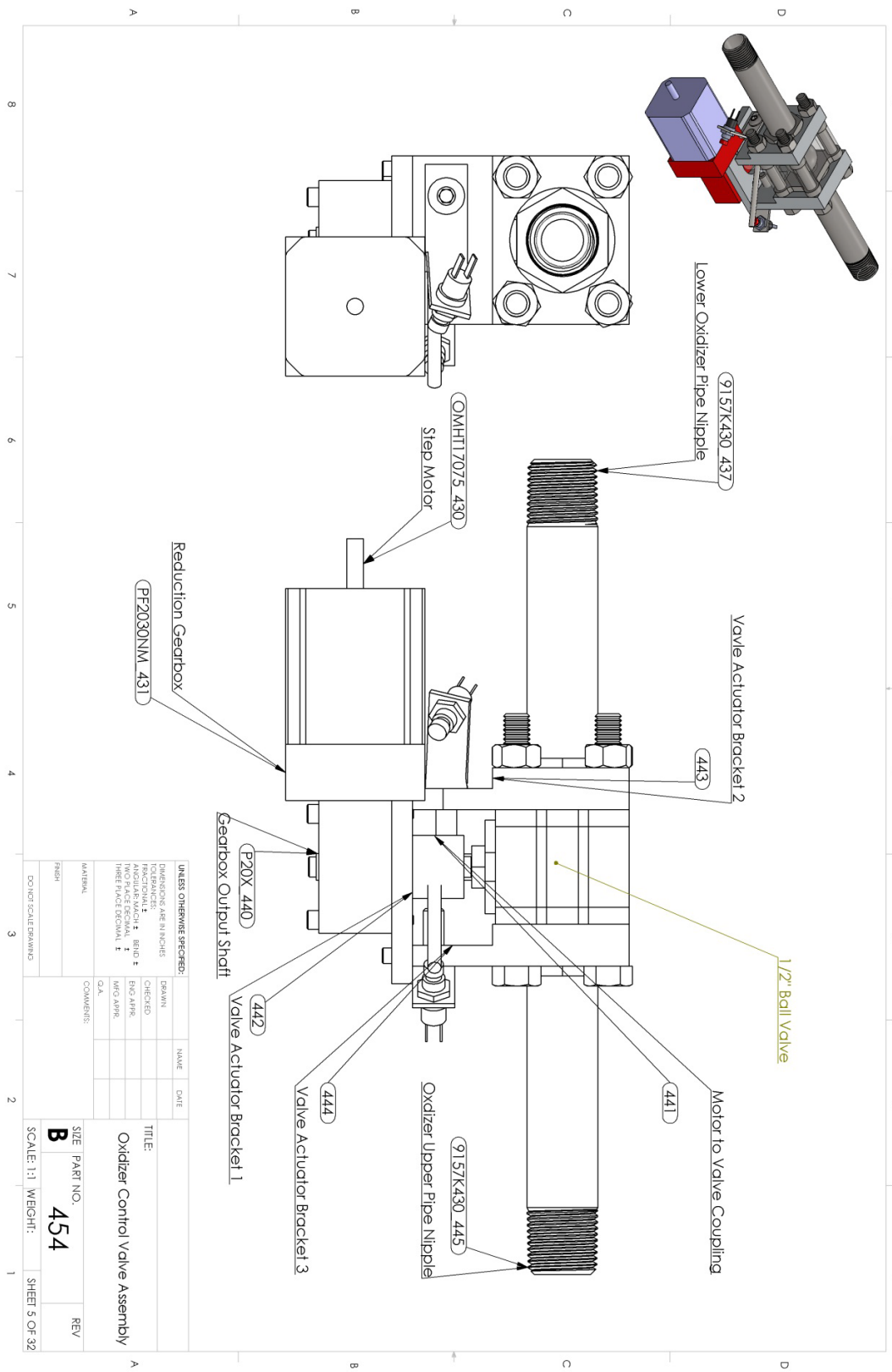


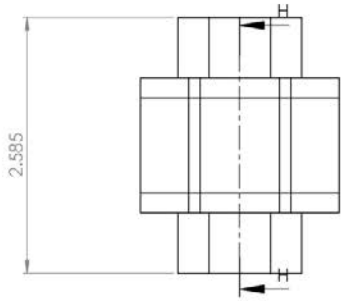
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
	CHECKED		
	ENG APPR.		
	MFG APPR.		
	Q.A.		
MATERIAL	COMMENTS:		
FINISH			
DO NOT SCALE DRAWING			

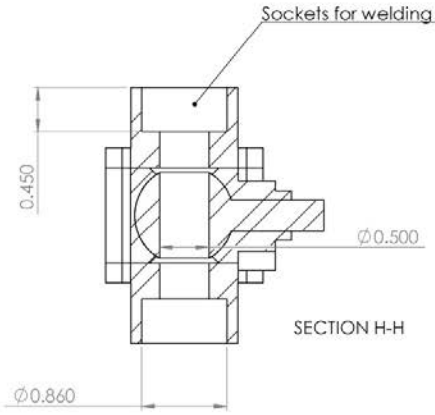
TITLE: Nozzle Assembly

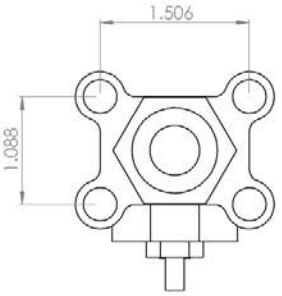
SIZE	PART NO.	REV
B	453	

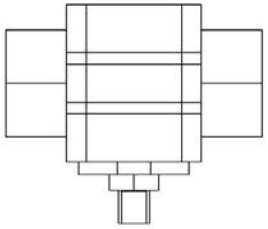
SCALE: 1:2 WEIGHT: SHEET 4 OF 32


1/2" Ball Valve
Lower Oxidizer Pipe Nipple
9157K430_437
Step Motor
OMHT17075_430
Reduction Gearbox
PF2030NM_431
Vavle Actuator Bracket 2
443
Gearbox Output Shaft
P20X_440
Valve Actuator Bracket 1
442
Valve Actuator Bracket 3
444
Motor to Valve Coupling
441
Oxdizer Upper Pipe Nipple
9157K430_445
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
MATERIAL
FINISH
DO NOT SCALE DRAWING
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NAME
DATE
TITLE:
Oxidizer Control Valve Assembly
SIZE
B
PART NO.
454
REV
SCALE: 1:1
WEIGHT:
SHEET 5 OF 32







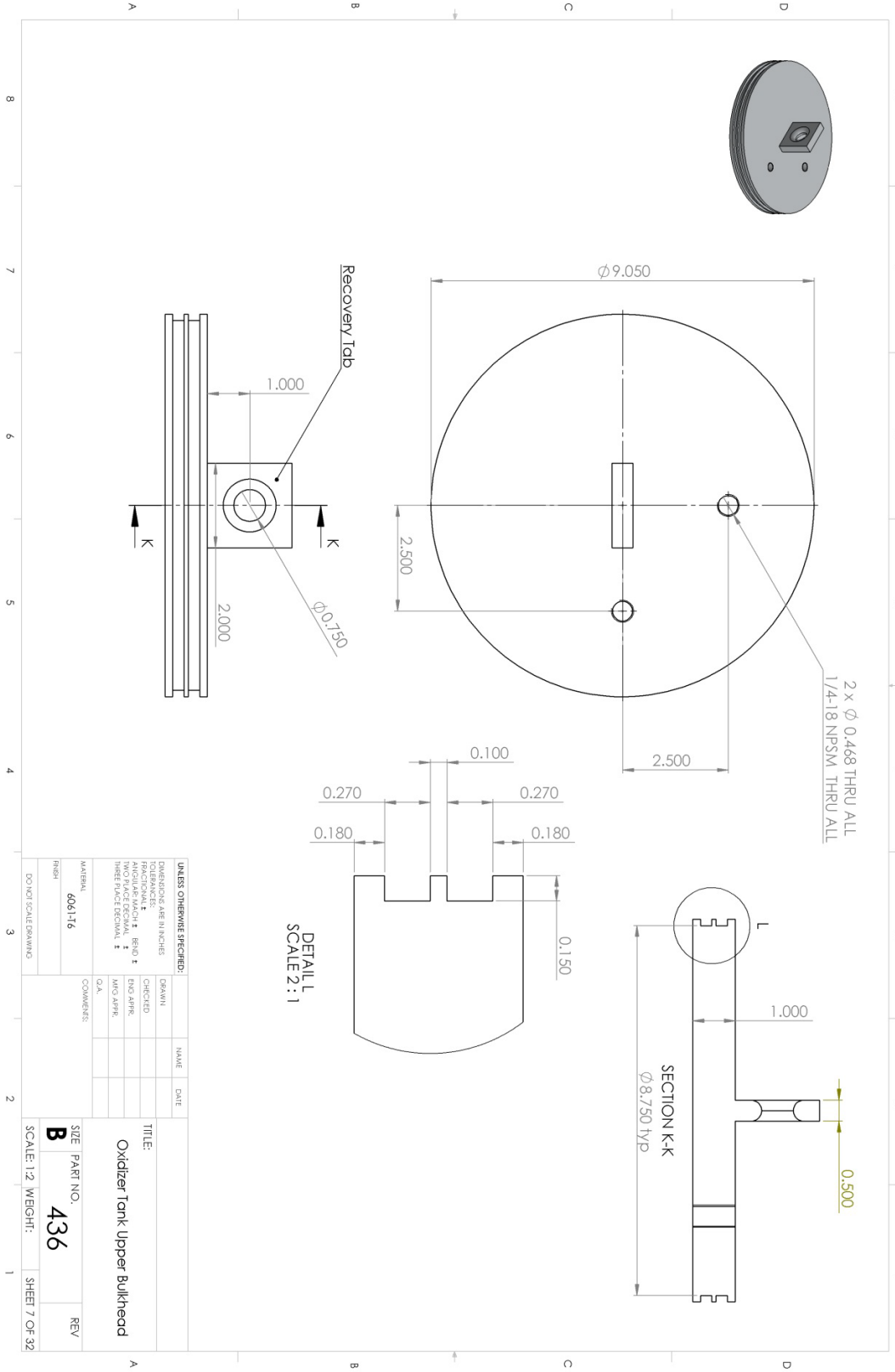


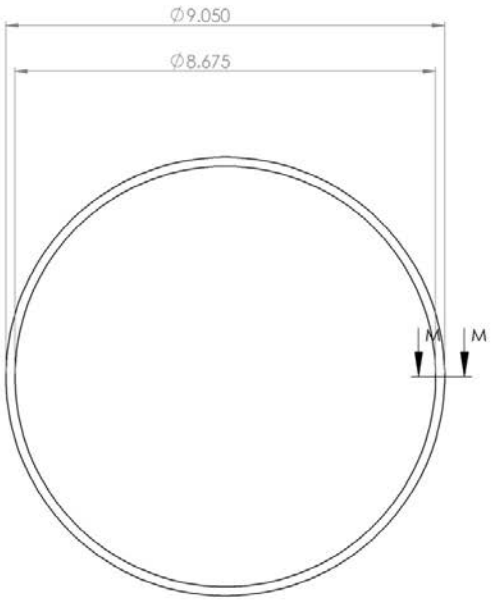
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
	CHECKED		
	ENG APPR.		
	MFG APPR.		
	Q.A.		
MATERIAL SS	COMMENTS: Manufactured by Watts Industries. Cv=13, all Stainless Steel Construction, PTFE seals, very similar to a Swagelok equivalent		
FINISH			
DO NOT SCALE DRAWING			

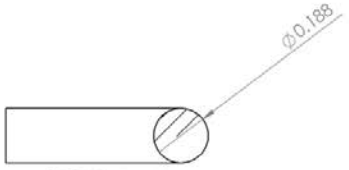
TITLE: Oxdizer Control Valve

SIZE	PART NO.	REV
B	455	

SCALE: 1:1 WEIGHT: SHEET 6 OF 32


Recovery Tab
1.000
2.000
K
K
Ø0.750
Ø9.050
2.500
2.500
2× Ø 0.468 THRU ALL
1/4-18 NPSM THRU ALL
0.100
0.270
0.270
0.180
0.180
0.150
DETAIL L
SCALE 2 : 1
L
1.000
0.500
SECTION K-K
Ø8.750 typ
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NAME
DATE
MATERIAL
6061-T6
FINISH
DO NOT SCALE DRAWING
TITLE:
Oxidizer Tank Upper Bulkhead
SIZE
B
PART NO.
436
REV
SCALE: 1:2
WEIGHT:
SHEET 7 OF 32





SECTION M-M
SCALE 3 : 1

APG
MSC #: 02049294
Order Qty of 1 = (1) Piece
Price: $5.89 ea
In stock: 6
Mfr: APG
Mfr #: S70371
Qty: ADD TO CART ADD TO LIST

Add To Cart Add to List
Email To A Friend Print this page

Specifications

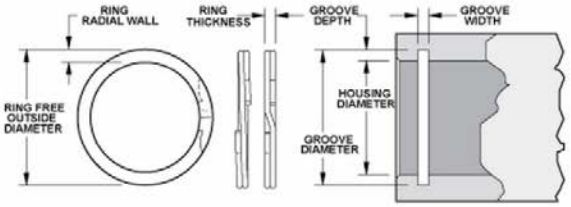
Item Number	02049294
Brand	APG
Cross Section Shape	Round
Material	Silicone
Dash Number	371
System of Measurement	Inch
Inside Diameter (Decimal Inch)	8.5000
Inside Diameter (Inch)	8-1/2
Outside Diameter (Decimal Inch)	8.8750
Outside Diameter (Inch)	8-7/8
Thickness (Inch)	3/16
Durometer (Shore A)	65 - 75
Maximum Temperature (F)	450.000
Minimum Temperature (F)	-80.000
Big Book Page #	4294

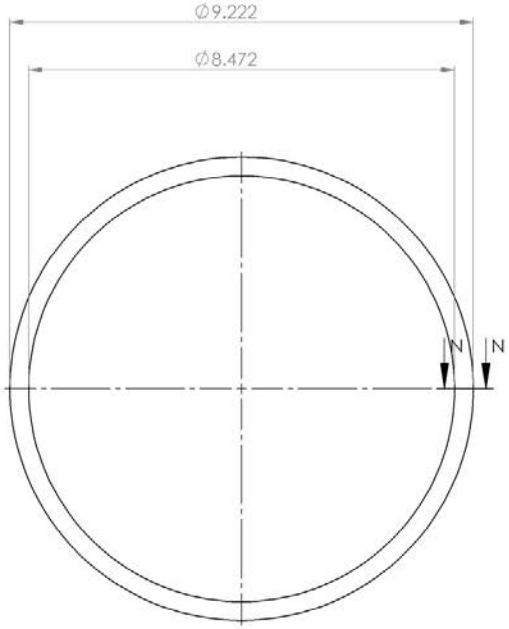
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
	CHECKED		
	ENG APPR.		
	MFG APPR.		
	Q.A.		
MATERIAL Silicon	COMMENTS:		
Finish			
DO NOT SCALE DRAWING			

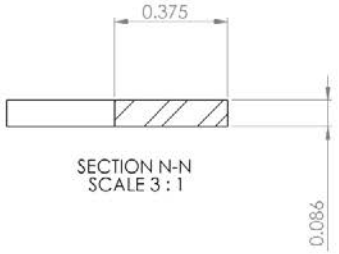
TITLE:
Oxidizer Tank 371-O-Ring

SIZE	PART NO.	REV
B	02049294_435	

SCALE: 1:2 WEIGHT: SHEET 8 OF 32







Part Number:	**WH-900**	
Ring Type:	**Internal Ring**	
Housing Diameter (in):	**9.000**	
Groove Diameter (in):	**9.222**	+/-.008
Groove Width (in):	**0.094**	+.006/-.000
Free Outside Diameter (in):	**9.293**	+.065/-.000
Ring Thickness (in):	**0.086**	+/-.004
Ring Radial Wall (in):	**0.375**	+.004/-.008
Number of Turns:	**2**	
Crimp:	**Yes**	
Ring Shear (lbs):	**102130**	(based on a safety factor of 3)
Groove Yield (lbs):	**70620**	(based on a groove material yield strength of 45,000 psi and a safety factor of 2)

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±

MATERIAL: Steel
FINISH:
DO NOT SCALE DRAWING

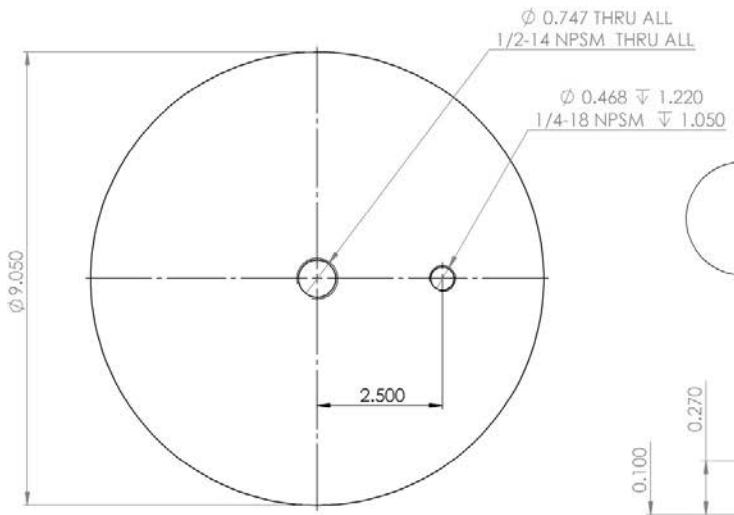
	NAME	DATE
DRAWN		
CHECKED		
ENG APPR.		
MFG APPR.		
Q.A.		

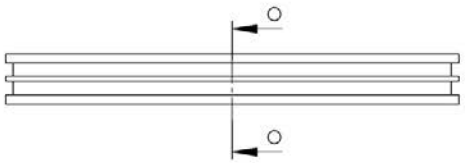
COMMENTS: Purchased from Smalley

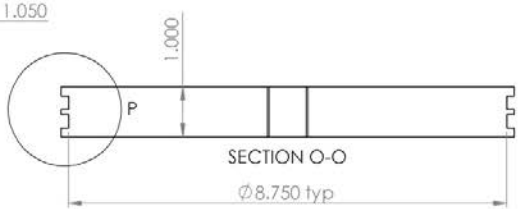
TITLE: Oxidizer Tank Snap Ring

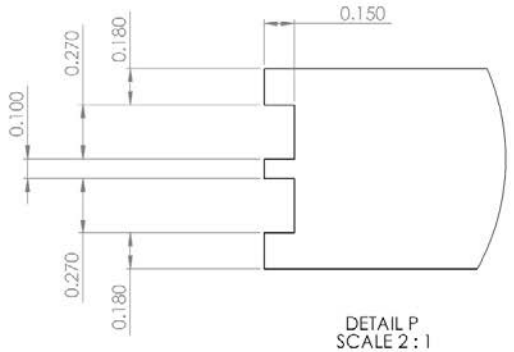
SIZE: B | PART NO.: WH900_433 | REV:

SCALE: 1:2 | WEIGHT: | SHEET 9 OF 32









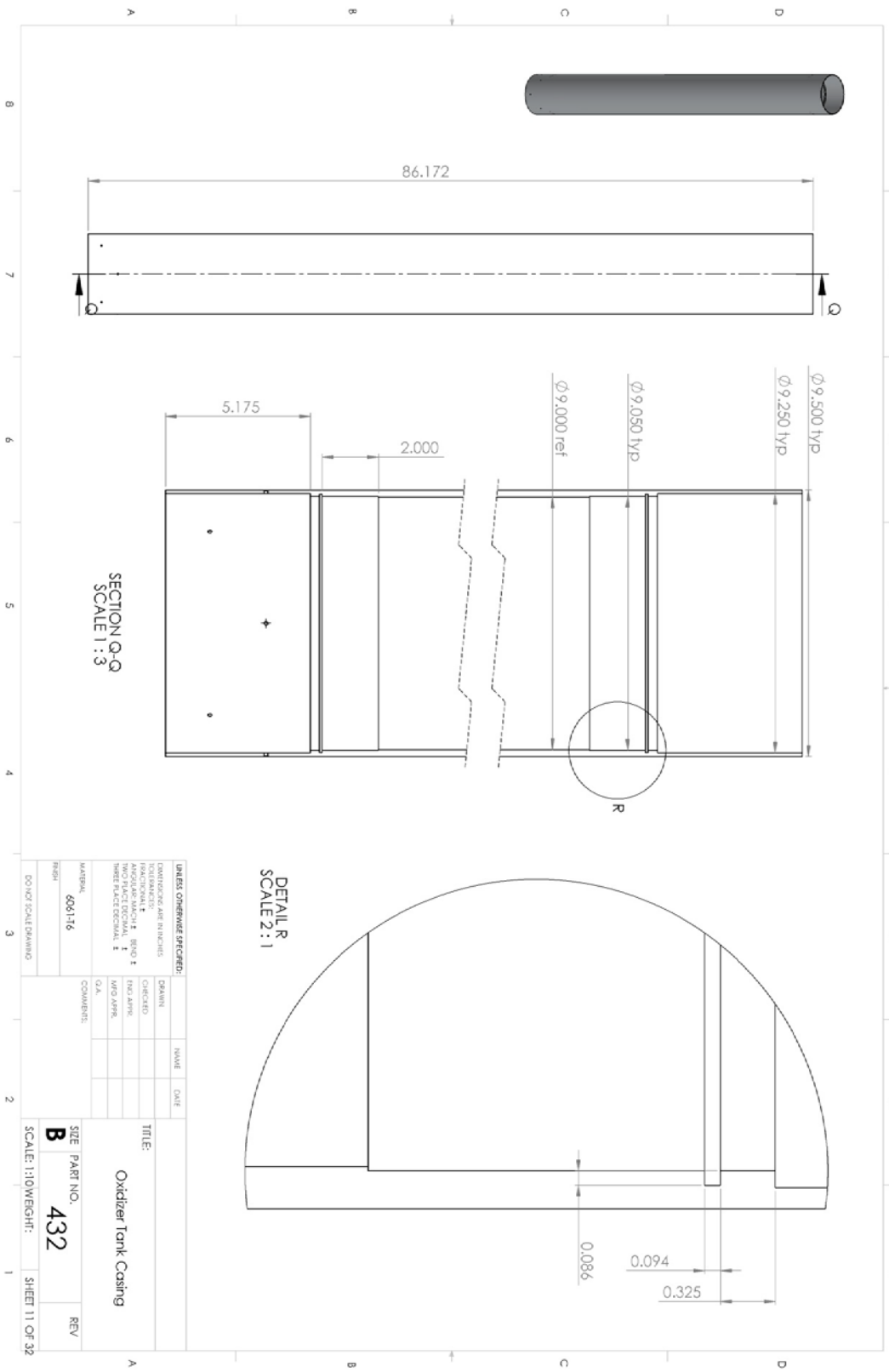
DETAIL P
SCALE 2 : 1

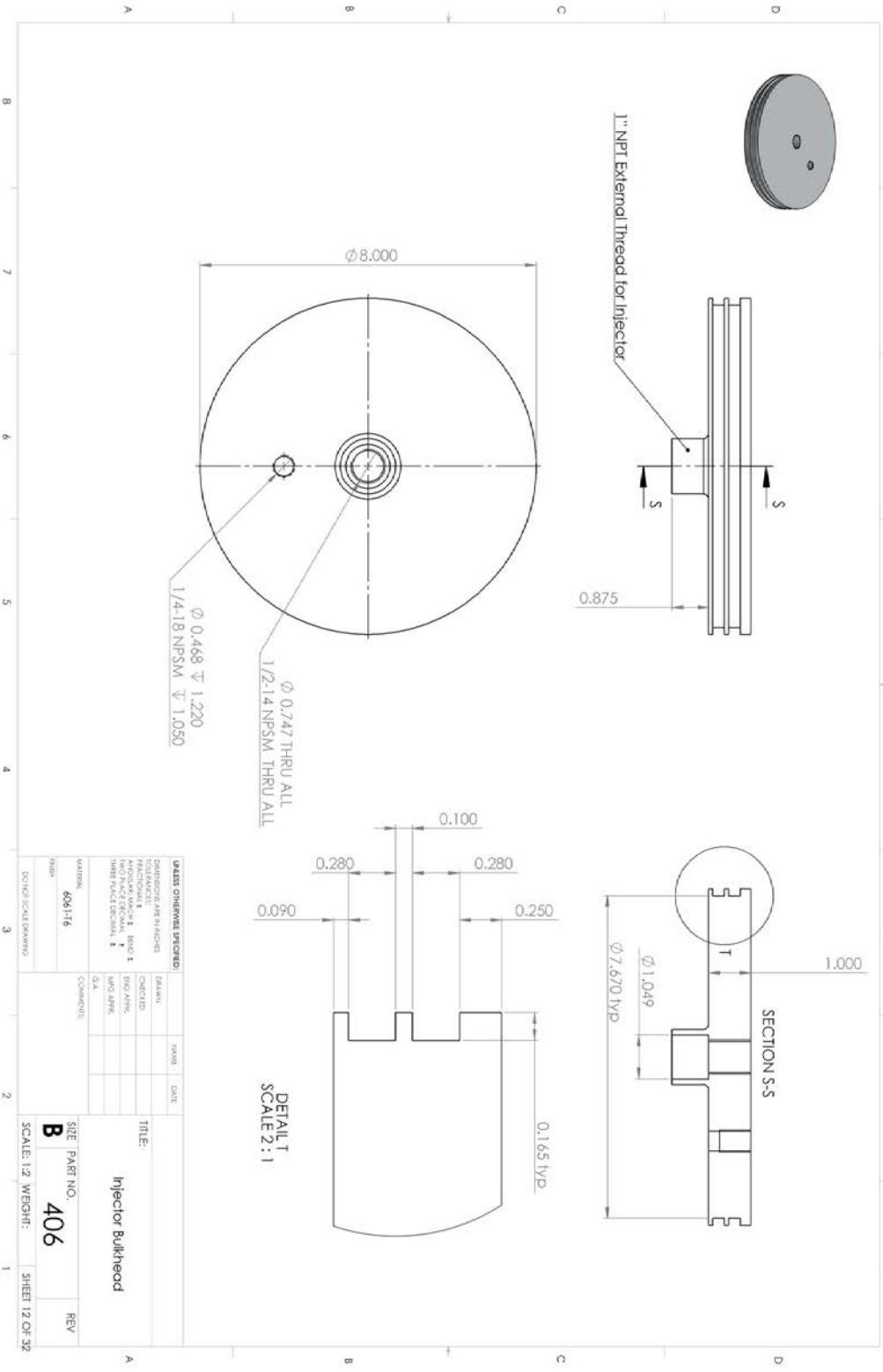
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
	CHECKED		
	ENG APPR.		
	MFG APPR.		
	Q.A.		
MATERIAL 6061-T6	COMMENTS:		
FINISH			
DO NOT SCALE DRAWING			

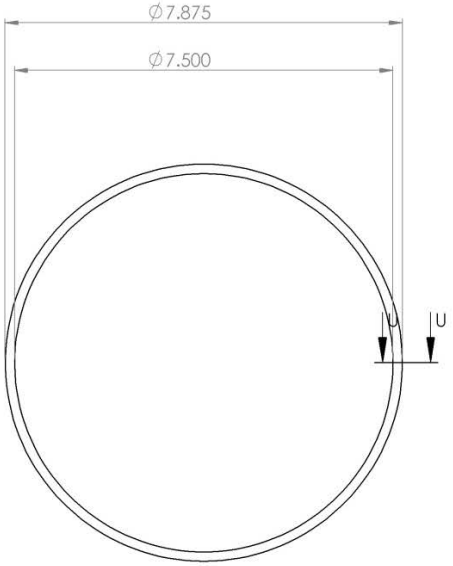
TITLE:
Oxidizer Lower Bulkhead

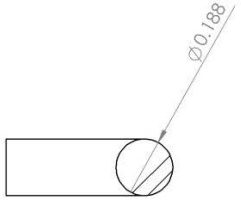
SIZE	PART NO.	REV
B	434	

SCALE: 1:2 WEIGHT: SHEET 10 OF 32


86.172
Q
Q
5.175
2.000
Ø9.000 ref
Ø9.050 typ
Ø9.250 typ
Ø9.500 typ
SECTION Q-Q
SCALE 1 : 3
R
DETAIL R
SCALE 2 : 1
0.086
0.094
0.325
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NAME
DATE
MATERIAL
6061-t6
FINISH
DO NOT SCALE DRAWING
TITLE:
Oxidizer Tank Casing
SIZE
B
PART NO.
432
REV
SCALE: 1:10
WEIGHT:
SHEET 11 OF 32


1" NPT External Thread for Injector
S
S
0.875
⌀8.000
⌀ 0.747 THRU ALL
1/2-14 NPSM THRU ALL
⌀ 0.468 ↧ 1.220
1/4-18 NPSM ↧ 1.050
0.100
0.280
0.280
0.090
0.250
0.165 typ
DETAIL T
SCALE 2 : 1
T
1.000
⌀1.049
⌀7.670 typ
SECTION S-S
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NAME
DATE
MATERIAL
6061-T6
FINISH
DO NOT SCALE DRAWING
TITLE:
Injector Bulkhead
SIZE
B
PART NO.
406
REV
SCALE: 1:2 WEIGHT:
SHEET 12 OF 32





SECTION U-U
SCALE 3 : 1

APG

MSC #: 02049252
Order Qty of 1 = (1) Piece
Price: $4.37 ea
In stock: 39
Mfr: APG
Mfr #: S70367

Qty: ADD TO CART ADD TO LIST

Add To Cart Add to List
Email To A Friend Print this page

Specifications

Item Number	02049252
Brand	APG
Cross Section Shape	Round
Material	Silicone
Dash Number	367
System of Measurement	Inch
Inside Diameter (Decimal Inch)	7.5000
Inside Diameter (Inch)	7-1/2
Outside Diameter (Decimal Inch)	7.8750
Outside Diameter (Inch)	7-7/8
Thickness (Inch)	3/16
Durometer (Shore A)	65 - 75
Maximum Temperature (F)	450.000
Minimum Temperature (F)	-80.000
Big Book Page #	4294

UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
	CHECKED		
	ENG APPR.		
	MFG APPR.		
	Q.A.		
MATERIAL Silicon	COMMENTS:		
FINISH			
DO NOT SCALE DRAWING			

TITLE:
Combustion Chamber 367-O-Ring

SIZE	PART NO.	REV
B	02049252_405	

SCALE: 1:2 WEIGHT: SHEET 13 OF 32

SS

Small Droplet Size Dense Fog

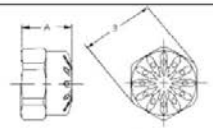
Metal

DESIGN FEATURES
- Multiple flat fan patterns
- Solid one-piece construction
- Female connection

SPRAY CHARACTERISTICS
- Relatively small droplets

Spray pattern: Dense Full Cone
Flow rates: 9.16 to 618 l/min
Spray angles:
SS4.8 thru **SS25** - 35°
SS35 thru **SS70** - 45°

Fog



Dimensions are approximate. Check with BETE for critical dimension applications.

SS Flow Rates and Dimensions
Full Cone, 3/4", 1" and 1-1/4" Pipe Size, BSP or NPT

Female Pipe Size	Nozzle Number	K Factor	LITERS PER MINUTE @ BAR 0.7 bar	1 bar	2 bar	3 bar	5 bar	10 bar	15 bar	Dimensions (mm) A	B	Wt (g)
3/4	SS4.8	10.9	9.16	10.8	15.5	19.0	24.5	34.6	42.4	25.4	35.1	85.1
	SS9	20.5	17.2	20.5	29.0	35.6	45.9	64.9	79.5			
	SS13	27.4	22.8	27.4	38.7	47.4	61.2	86.5	106			
	SS16	33.1	34.3	41.1	58.1	71.1	91.8	130	159			
1	SS25	57.0	47.7	57.0	80.6	98.8	127	180	221	36.5	42.2	140
	SS35	79.8	66.8	79.8	113	138	178	252	309			
1 1/4	SS50	114	95.4	114	161	198	255	361	442	51.8	53.1	227
	SS70	160	134	160	226	277	357	505	618			

Flow Rate (l/min) = $K\sqrt{bar}$

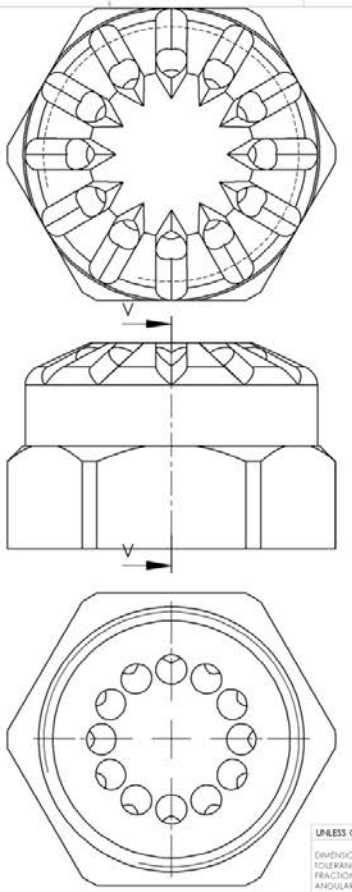
Standard Materials: Brass, 303 and 316 Stainless Steel.

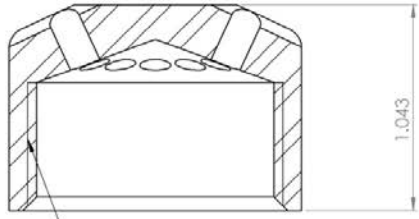
SPECIAL PURPOSE

TO ORDER: specify pipe size, connection type, nozzle number, spray angle, and material

106

www.BETE.com





SECTION V-V

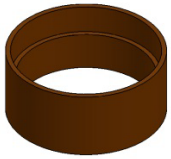
1" Female NPT

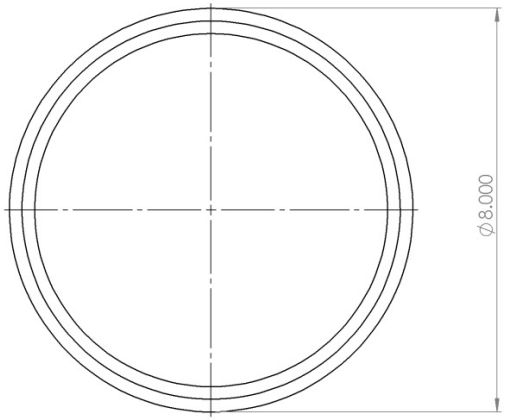
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES	DRAWN		
TOLERANCES: FRACTIONAL ±	CHECKED		
ANGULAR: MACH ± BEND ±	ENG APPR.		
TWO PLACE DECIMAL ±	MFG APPR.		
THREE PLACE DECIMAL ±	Q.A.		
MATERIAL: 304 SS	COMMENTS:		
FINISH			
DO NOT SCALE DRAWING			

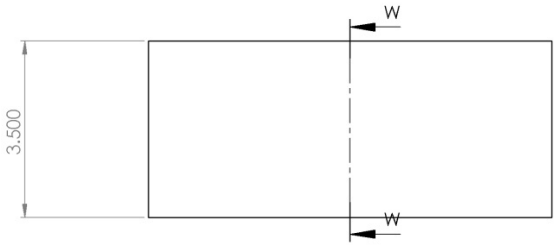
TITLE: Injector

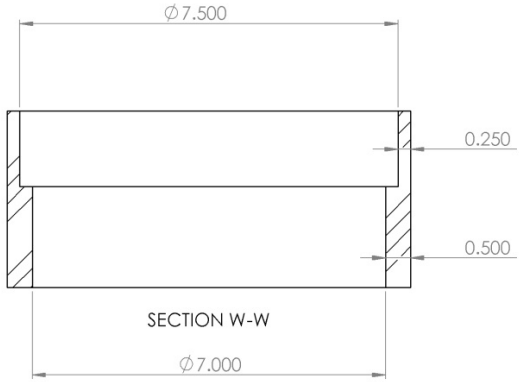
SIZE	PART NO.	REV
B	SS25_410	

SCALE: 2:1 | WEIGHT: | SHEET 14 OF 32







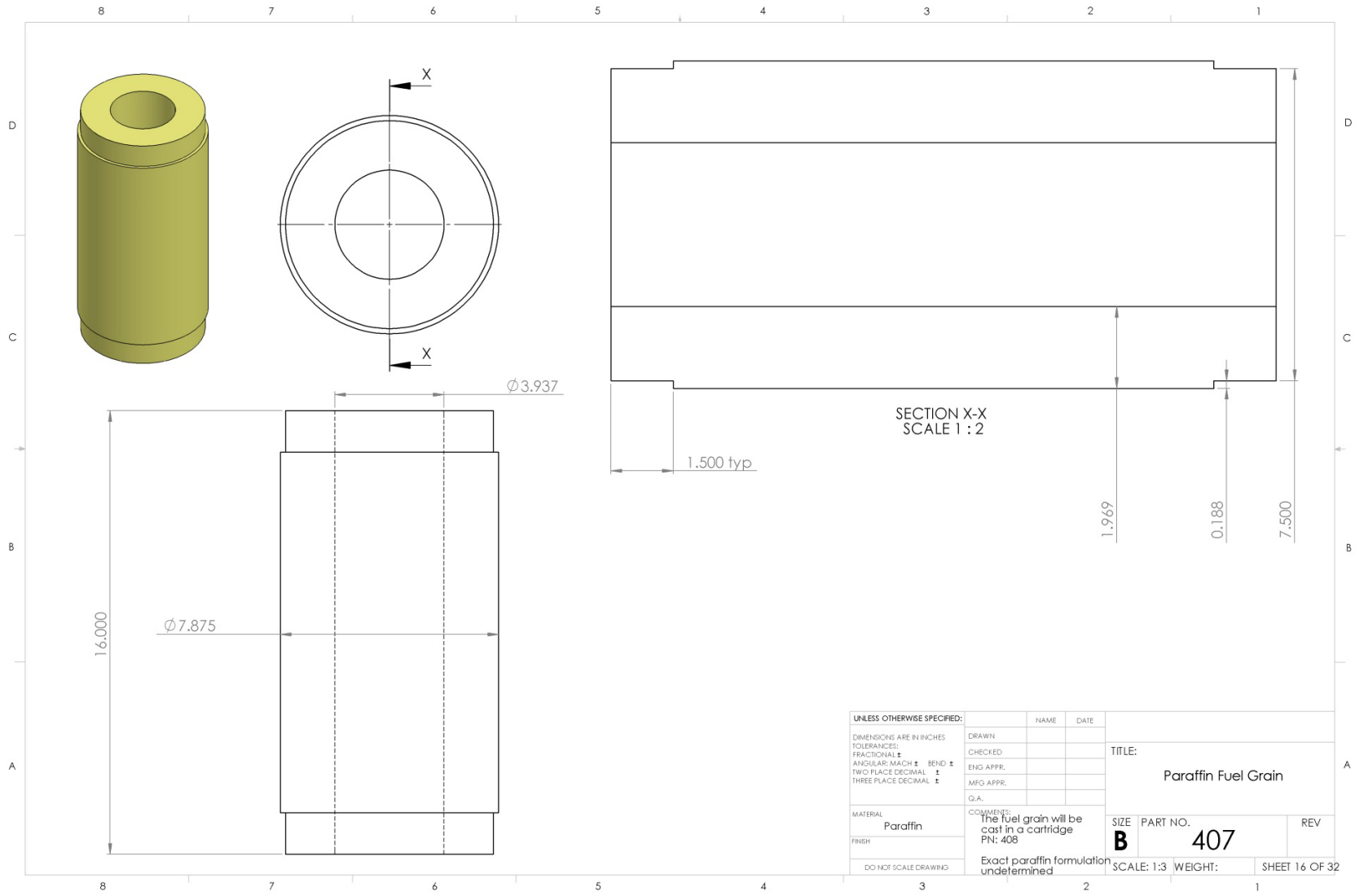


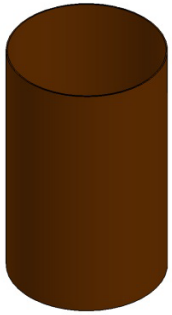
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
	CHECKED		
	ENG APPR.		
	MFG APPR.		
	Q.A.		
MATERIAL Canvas Phenolic	COMMENTS:		
FINISH			
DO NOT SCALE DRAWING			

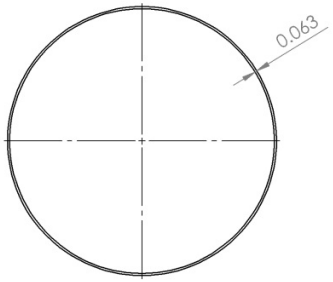
TITLE:
Chamber Pre-Burner Insulator

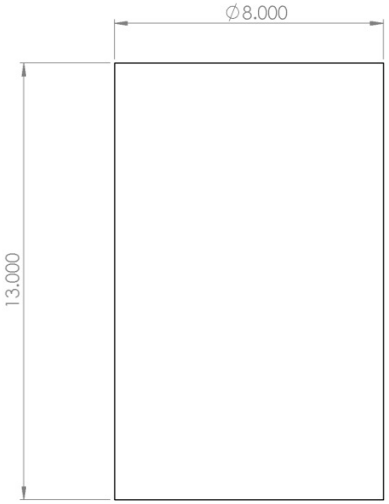
SIZE	PART NO.	REV
B	446	

SCALE: 1:2 WEIGHT: SHEET 15 OF 32


X
X
SECTION X-X
SCALE 1 : 2
1.500 typ
1.969
0.188
7.500
⌀3.937
⌀7.875
16.000
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
MATERIAL
Paraffin
FINISH
DO NOT SCALE DRAWING
COMMENTS:
The fuel grain will be cast in a cartridge
PN: 408
Exact paraffin formulation undetermined
TITLE:
Paraffin Fuel Grain
SIZE
B
PART NO.
407
REV
SCALE: 1:3
WEIGHT:
SHEET 16 OF 32





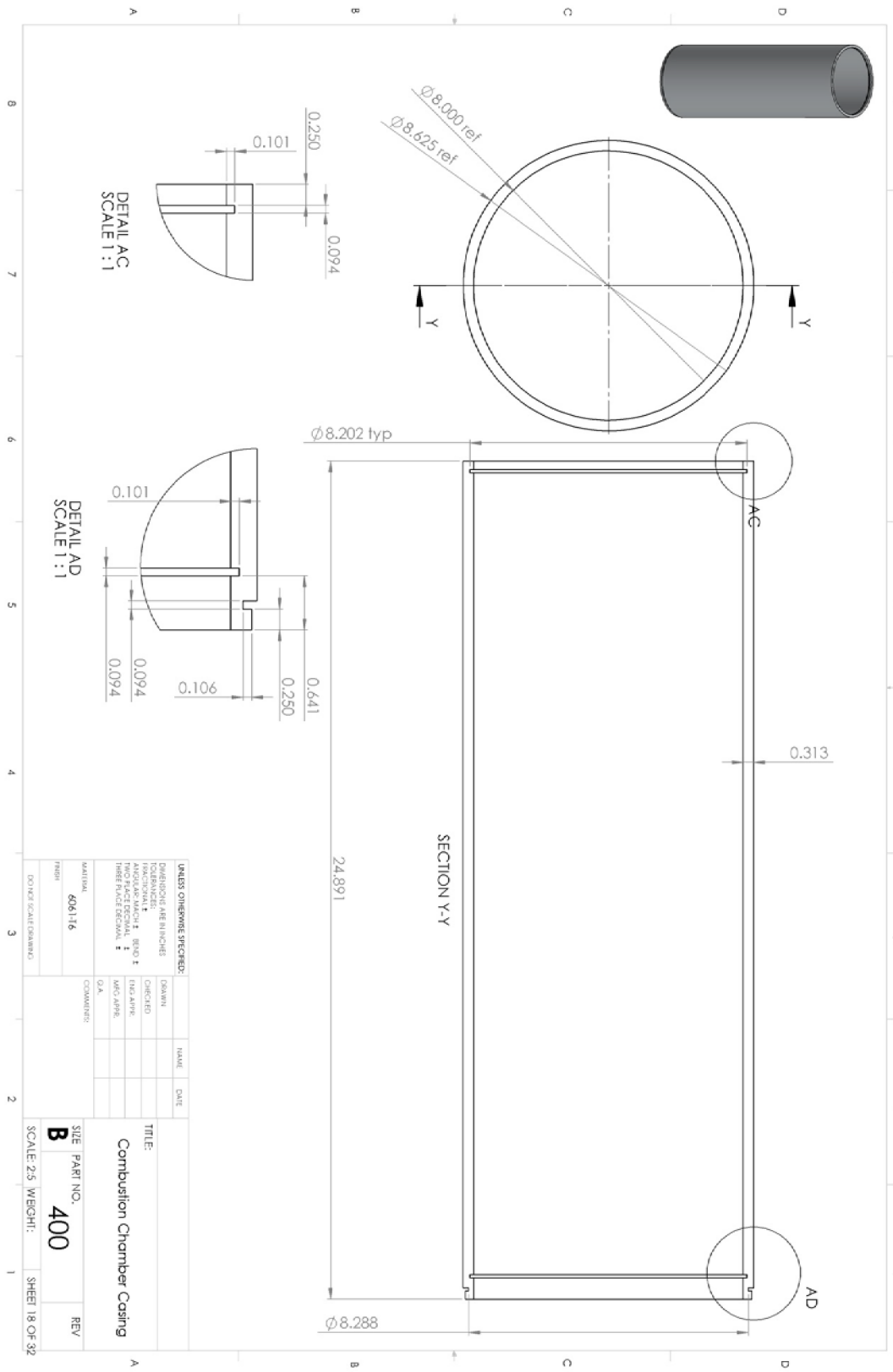


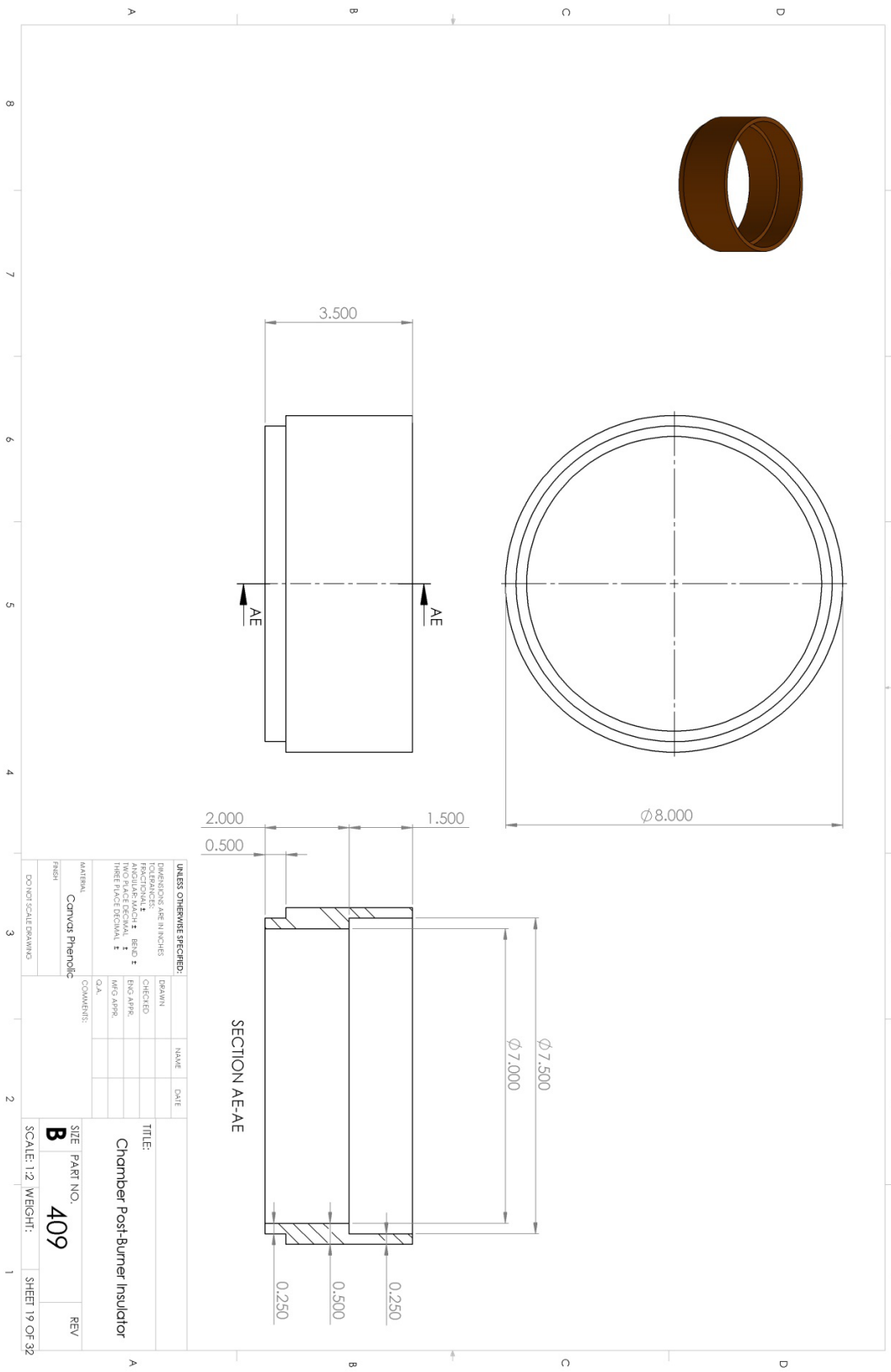
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
	CHECKED		
	ENG APPR.		
	MFG APPR.		
	Q.A.		
MATERIAL: Paper Phenolic	COMMENTS:		
FINISH			
DO NOT SCALE DRAWING			

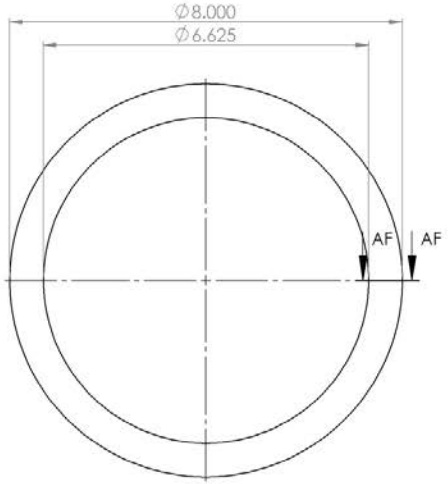
TITLE: Grain Insulator/Cartridge

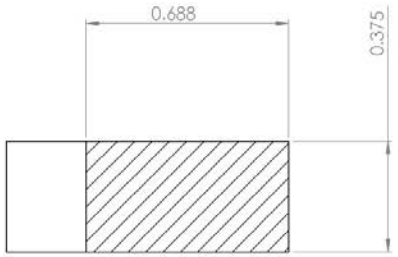
SIZE	PART NO.	REV
B	408	

SCALE: 1:3 WEIGHT: SHEET 17 OF 32


Ø8.000 ref
Ø8.625 ref
Y
Y
0.101
0.250
0.094
DETAIL AC
SCALE 1 : 1
Ø8.202 typ
AC
0.313
SECTION Y-Y
24.891
AD
Ø8.288
0.101
0.094
0.094
0.106
0.250
0.641
DETAIL AD
SCALE 1 : 1
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NAME
DATE
MATERIAL
6061-T6
FINISH
DO NOT SCALE DRAWING
TITLE:
Combustion Chamber Casing
SIZE
B
PART NO.
400
REV
SCALE: 2:5 WEIGHT:
SHEET 18 OF 32


3.500
AE
AE
⌀8.000
2.000
1.500
0.500
SECTION AE-AE
⌀7.500
⌀7.000
0.250
0.500
0.250
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NAME
DATE
MATERIAL
Canvas Phenolic
FINISH
DO NOT SCALE DRAWING
TITLE:
Chamber Post-Burner Insulator
SIZE
B
PART NO.
409
REV
SCALE: 1:2
WEIGHT:
SHEET 19 OF 32





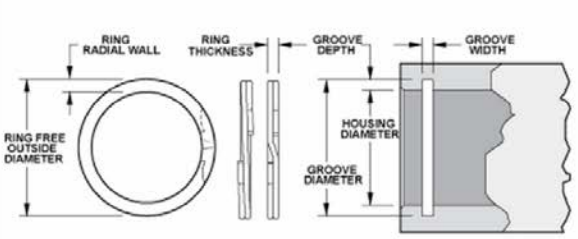
SECTION AF-AF
SCALE 3 : 1

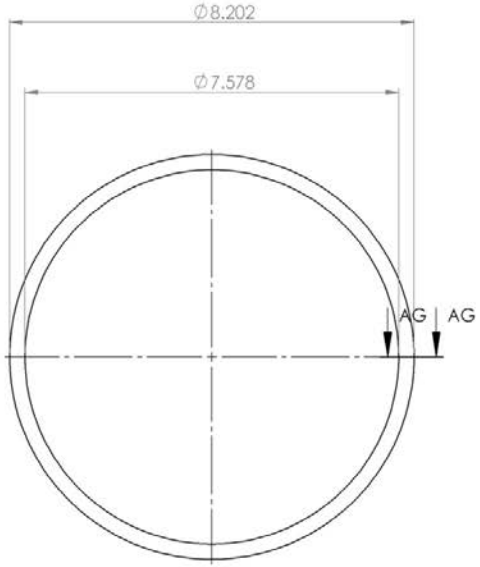
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
	CHECKED		
	ENG APPR.		
	MFG APPR.		
	Q.A.		
MATERIAL 6061-T6	COMMENTS:		
FINISH			
DO NOT SCALE DRAWING			

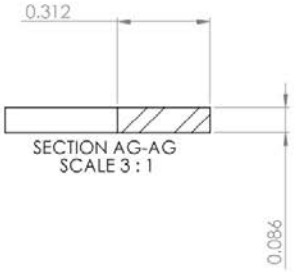
TITLE:
Nozzle Retainer

SIZE	PART NO.	REV
B	404	

SCALE: 1:2 WEIGHT: SHEET 20 OF 32







Part Number:	**WH-800**	
Ring Type:	**Internal Ring**	
Housing Diameter (in):	**8.000**	
Groove Diameter (in):	**8.202**	+/-.008
Groove Width (in):	**0.094**	+.006/-.000
Free Outside Diameter (in):	**8.266**	+.065/-.000
Ring Thickness (in):	**0.086**	+/-.004
Ring Radial Wall (in):	**0.312**	+.004/-.008
Number of Turns:	**2**	
Crimp:	**Yes**	
Ring Shear (lbs):	**90780**	(based on a safety factor of 3)
Groove Yield (lbs):	**57110**	(based on a groove material yield strength of 45,000 psi and a safety factor of 2)

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±

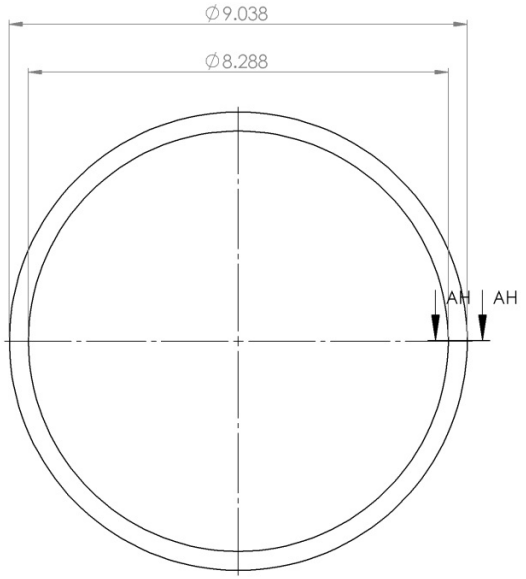
	NAME	DATE
DRAWN		
CHECKED		
ENG APPR.		
MFG APPR.		
Q.A.		
COMMENTS:		

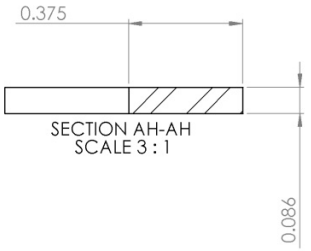
MATERIAL: Steel
Finish
DO NOT SCALE DRAWING

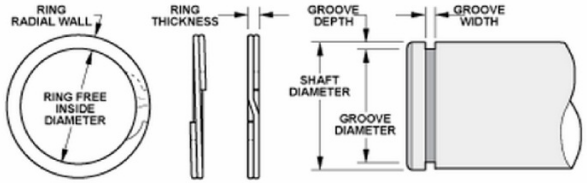
TITLE: Combustion Chamber Snap Ring

SIZE B | PART NO. WH800_403 | REV

SCALE: 1:2 | WEIGHT: | SHEET 21 OF 32







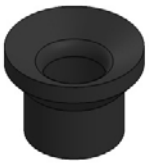
Part Number:	**WS-850**	
Ring Type:	**External Ring**	
Shaft Diameter (in):	**8.500**	
Groove Diameter (in):	**8.288**	+/-.008
Groove Width (in):	**0.094**	+.006/-.000
Free Inside Diameter (in):	**8.220**	+.000/-.070
Ring Thickness (in):	**0.086**	+/-.004
Ring Radial Wall (in):	**0.375**	+.004/-.008
Number of Turns:	**2**	
Crimp:	**Yes**	
Ring Shear (lbs):	**96450** (based on a safety factor of 3)	
Groove Yield (lbs):	**63690** (based on a groove material yield strength of 45,000 psi and a safety factor of 2)	

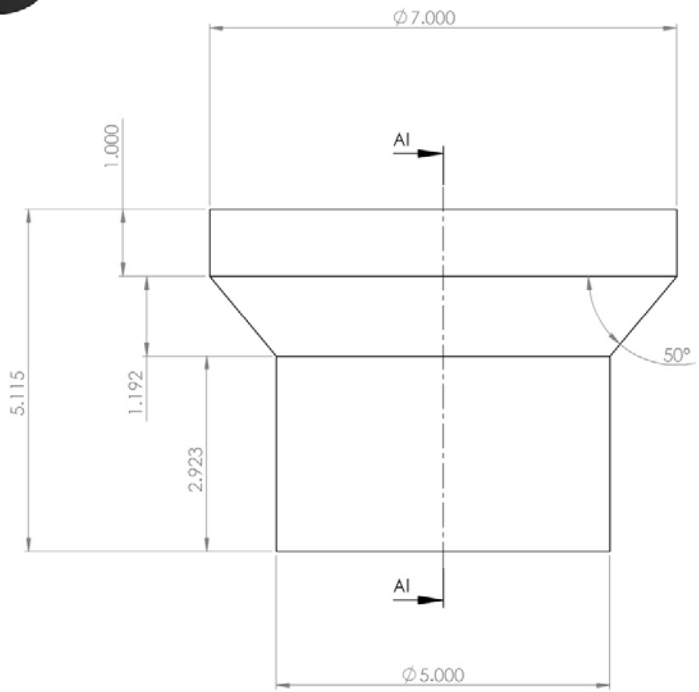
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
	CHECKED		
	ENG APPR.		
	MFG APPR.		
	Q.A.		
MATERIAL Steel	COMMENTS:		
FINISH			
DO NOT SCALE DRAWING			

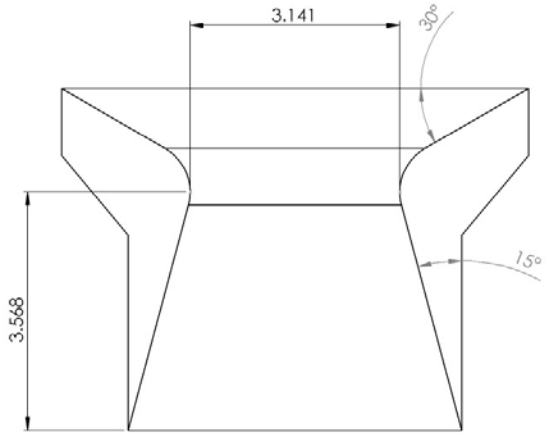
TITLE:
External Snap Ring

SIZE	PART NO.	REV
B	WS850_447	

SCALE: 1:2 WEIGHT: SHEET 22 OF 32







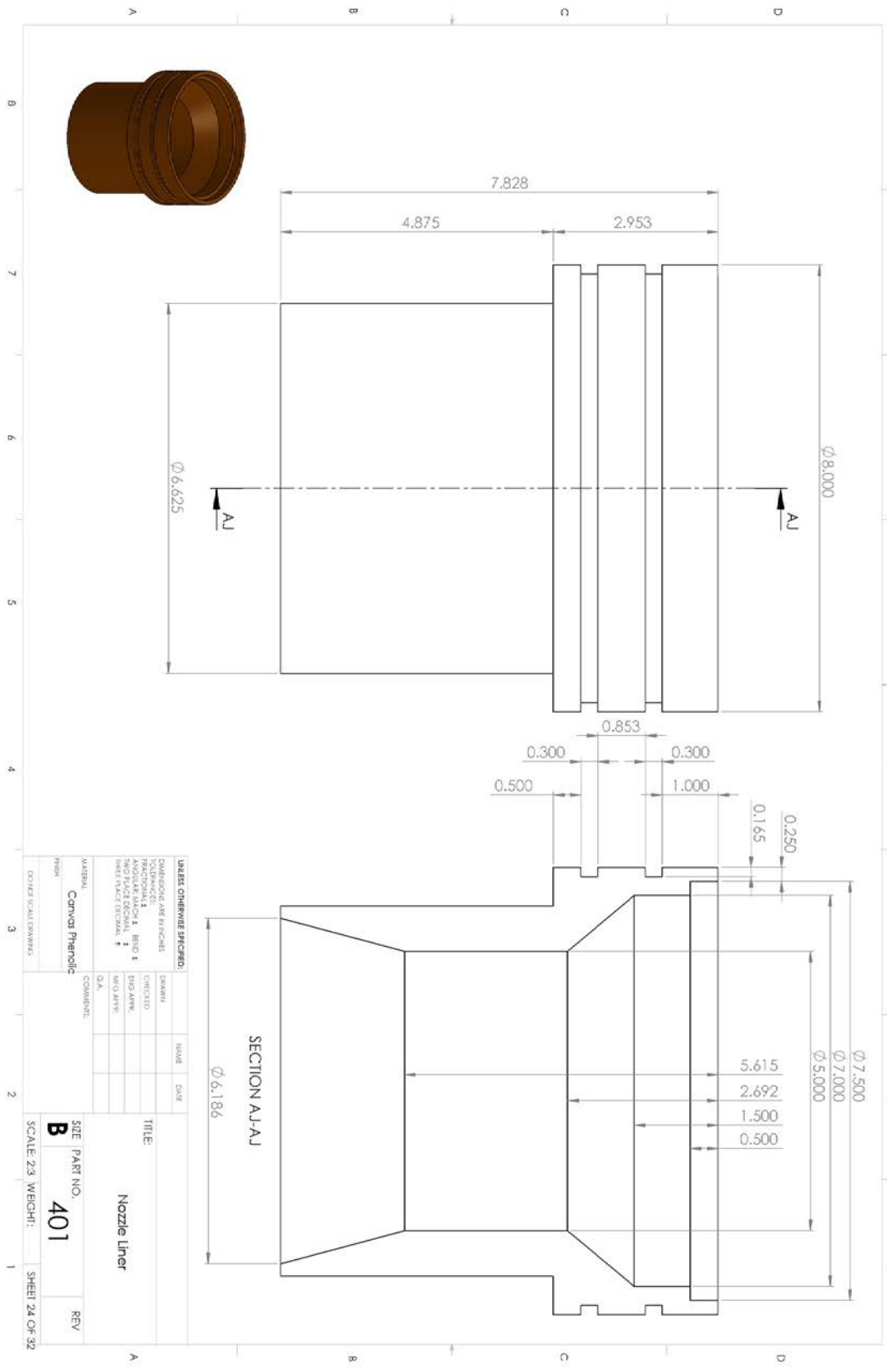
SECTION A1-A1

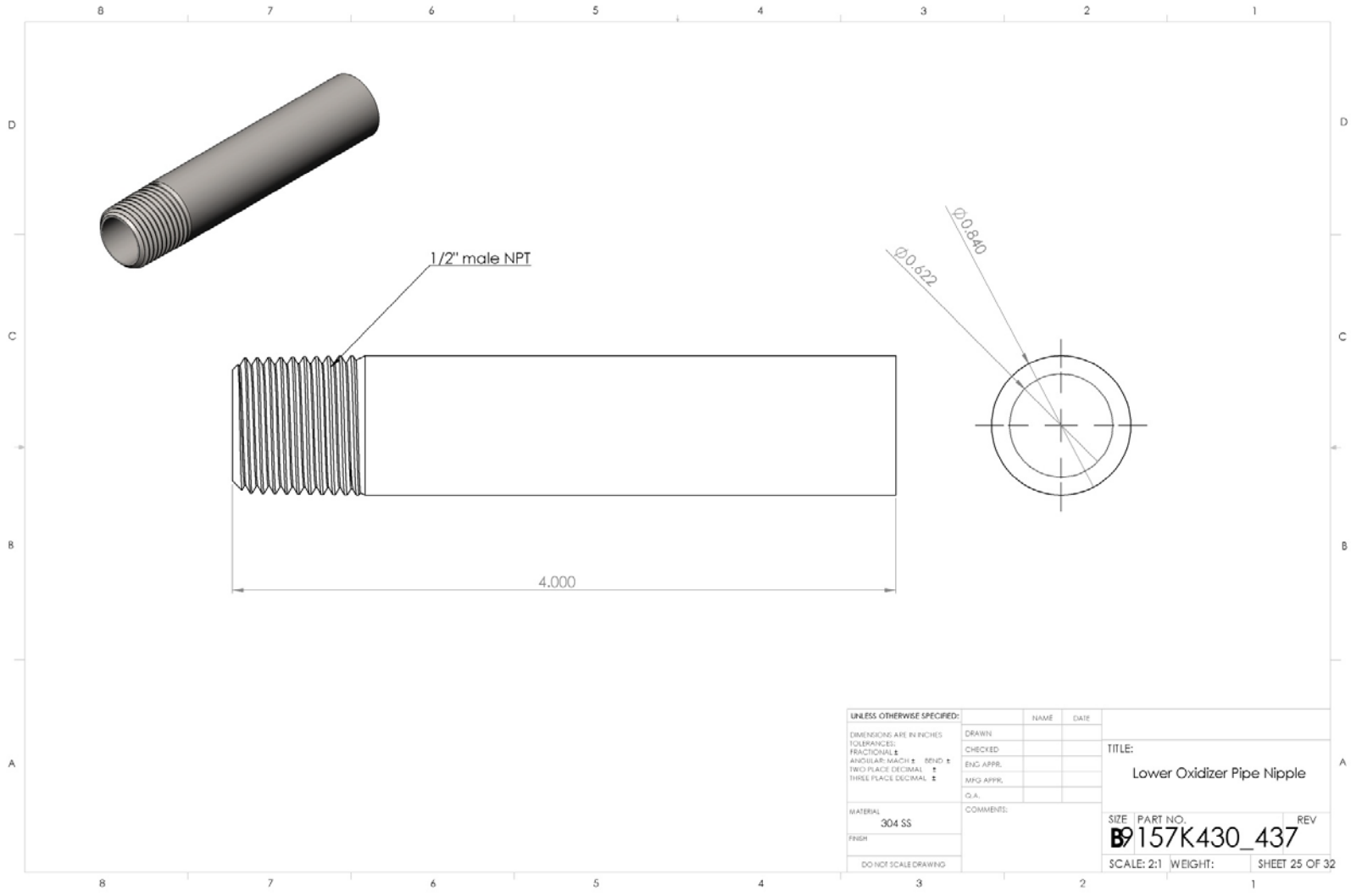
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
	CHECKED		
	ENG APPR.		
	MFG APPR.		
	Q.A.		
MATERIAL Fine Grain Graphite	COMMENTS:		
FINISH			
DO NOT SCALE DRAWING			

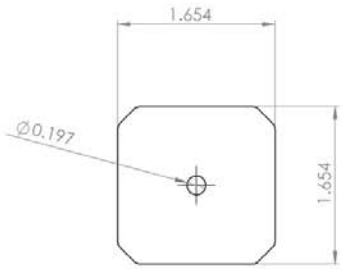
TITLE: Nozzle Throat

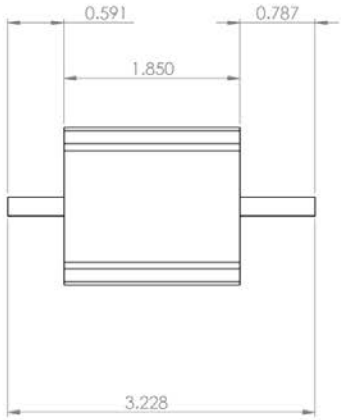
SIZE	PART NO.	REV
B	447	

SCALE: 2:3 WEIGHT: SHEET 23 OF 32


7.828
4.875
2.953
Ø6.625
Ø8.000
A
A
0.853
0.300
0.300
0.500
1.000
0.165
0.250
SECTION A-A
Ø6.186
Ø7.500
Ø7.000
Ø5.000
5.615
2.692
1.500
0.500
UNLESS OTHERWISE SPECIFIED:
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NAME
DATE
MATERIAL
Canvas Phenolic
FINISH
DO NOT SCALE DRAWING
TITLE:
Nozzle Liner
SIZE
B
PART NO.
401
REV
SCALE: 2:3
WEIGHT:
SHEET 24 OF 32


1/2" male NPT
Ø0.840
Ø0.622
4.000
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
MATERIAL
304 SS
FINISH
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
Lower Oxidizer Pipe Nipple
SIZE
PART NO.
REV
B
157K430_437
SCALE: 2:1
WEIGHT:
SHEET 25 OF 32



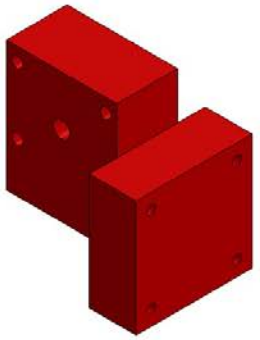


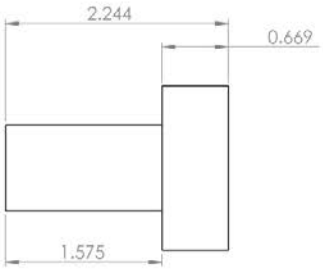
RECOMMENDED MOTORS

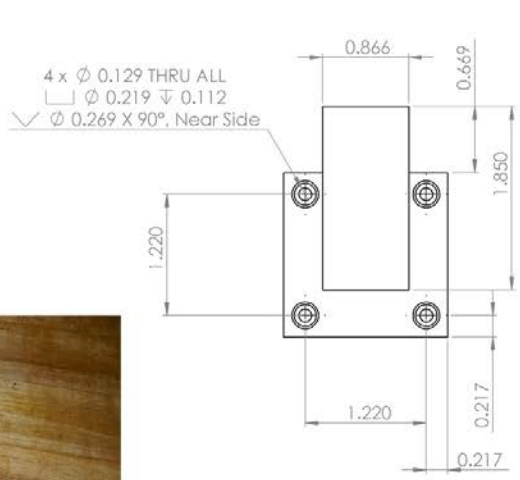
MODEL NO.	MOTOR CONNECTION 1 = SERIES 2 = PARALLEL 3 = UNIPOLAR	MOTOR LENGTH mm (in)	MAXIMUM HOLDING TORQUE¹ (oz-in)	LEADS	STEP ANGLE (DEG)	VOLTS	AMPS	OHMS	MH	ROTOR INERTIA (oz-in²/ g-cm²)	MOTOR WEIGHT g (lb)
OMHT11-013	2	48 (1.87)	15	4	1.8	2.0	1.0	2.0	2.6	0.098/18	177 (0.39)
OM5014-042	2	40 (1.57)	26.0	4	1.8	4.8	1.0	4.2	5.5	0.109/20	213 (0.47)
OMHT17-075	1	47 (1.85)	62.8	8	1.8	5.7	0.85	6.6	12.0	0.37/68	331 (0.73)
	2					2.8	1.70	1.7	3.0		
	3		44.4			4.0	1.20	3.3	3.0		
OMHT17-275	1	48 (1.90)	62.3	8	1.8	5.7	0.85	6.6	10.0	0.44/82	357 (0.79)
	2					2.8	1.70	1.7	2.5		
	3		44.0			4.0	1.20	3.3	2.5		
	1		76.6			7.4	0.71	1.7	21.6		454

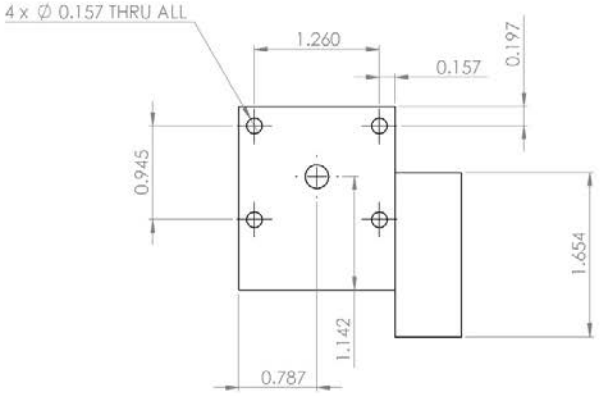
UNLESS OTHERWISE SPECIFIED: NAME DATE
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
MATERIAL
FINISH
DO NOT SCALE DRAWING
COMMENTS: Standard NEMA 17 Mount

TITLE: Step Motor
SIZE B | PART NO. OMHT17075_430 | REV
SCALE: 1:1 | WEIGHT: | SHEET 26 OF 32







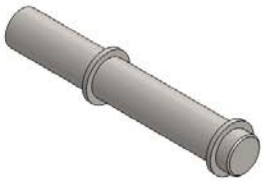


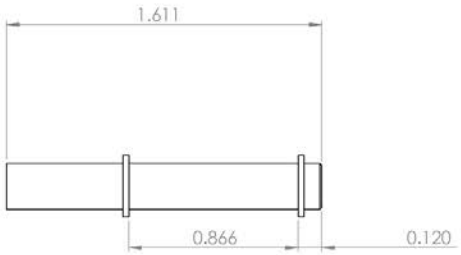
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
	CHECKED		
	ENG APPR.		
	MFG APPR.		
	Q.A.		
MATERIAL 6061-T6	COMMENTS: 30:1 Reduction		
FINISH			
DO NOT SCALE DRAWING			

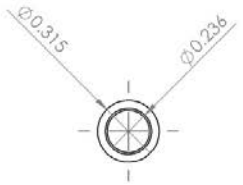
TITLE:
Reduction Gearbox

SIZE | PART NO. | REV
BF2030NM_431

SCALE: 1:1 WEIGHT: | SHEET 27 OF 32







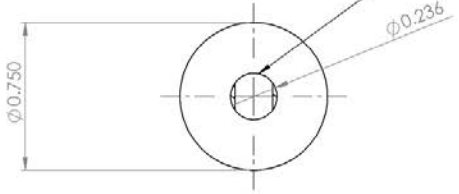
		NAME	DATE
UNLESS OTHERWISE SPECIFIED:			
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
	CHECKED		
	ENG APPR.		
	MFG APPR.		
	Q.A.		
	COMMENTS:		
MATERIAL Steel			
FINISH			
DO NOT SCALE DRAWING			

TITLE:
Gearbox Output Shaft

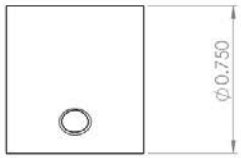
SIZE	PART NO.	REV
B	P20X_440	

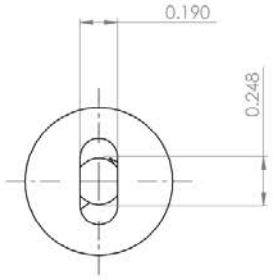
SCALE: 2:1 WEIGHT: SHEET 28 OF 32

Necessary Keyway Dimensions Undetermined



Gear Box End





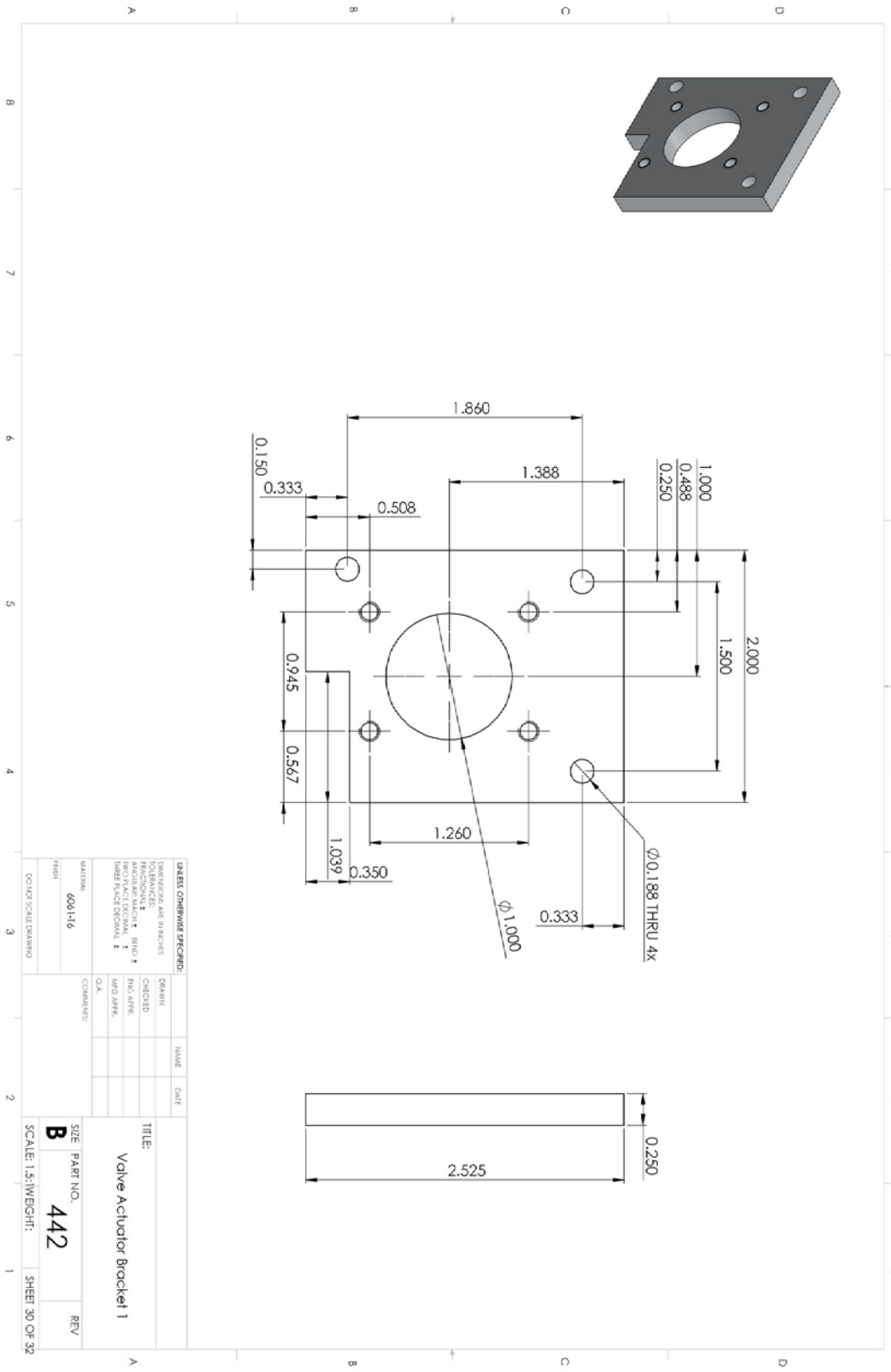
Valve End

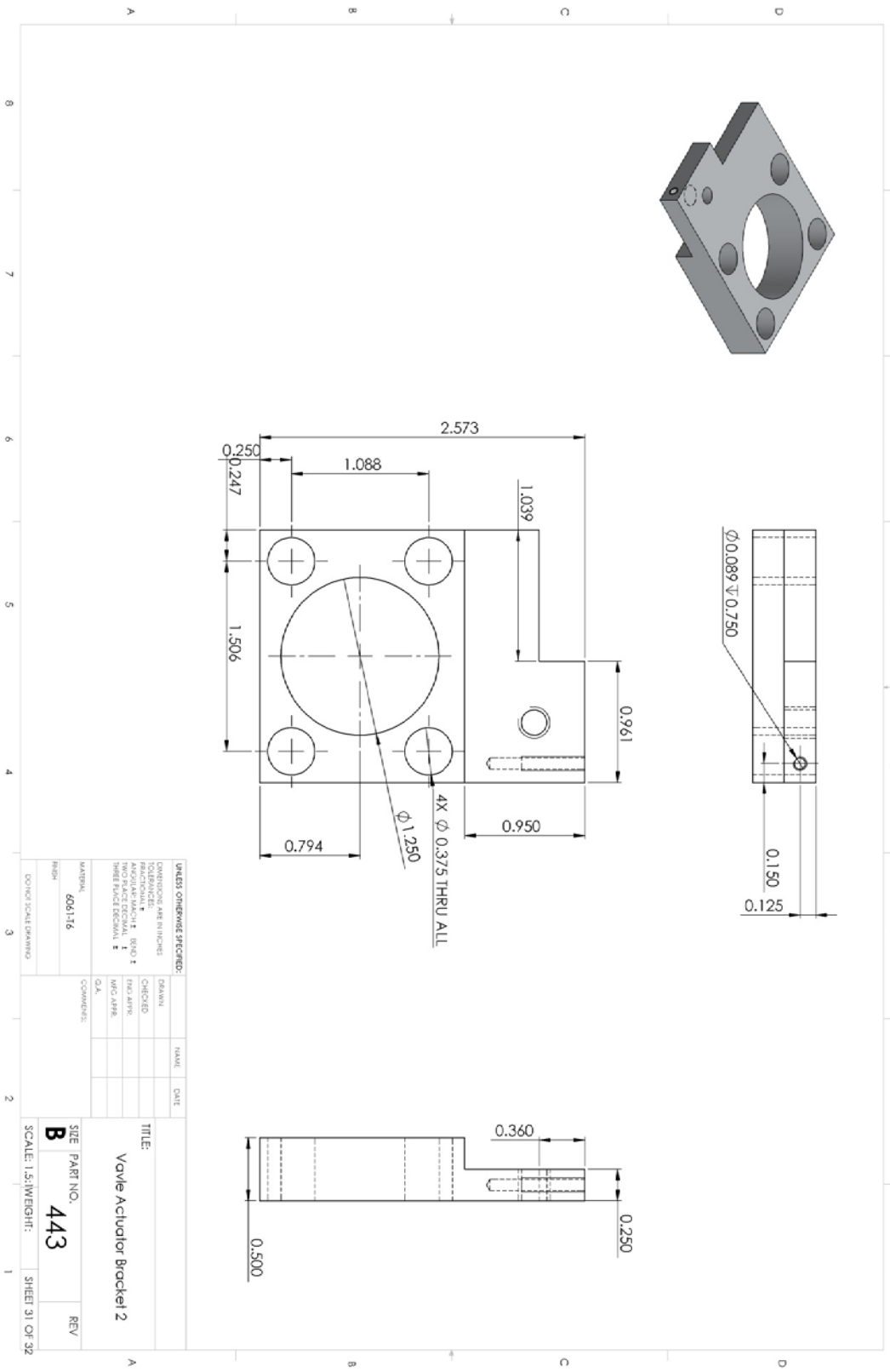
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
	CHECKED		
	ENG APPR.		
	MFG APPR.		
	Q.A.		
	COMMENTS:		
MATERIAL Steel			
FINISH			
DO NOT SCALE DRAWING			

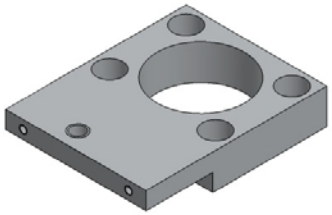
TITLE:
Motor to Valve Coupling

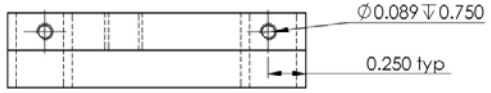
SIZE	PART NO.	REV
B	441	

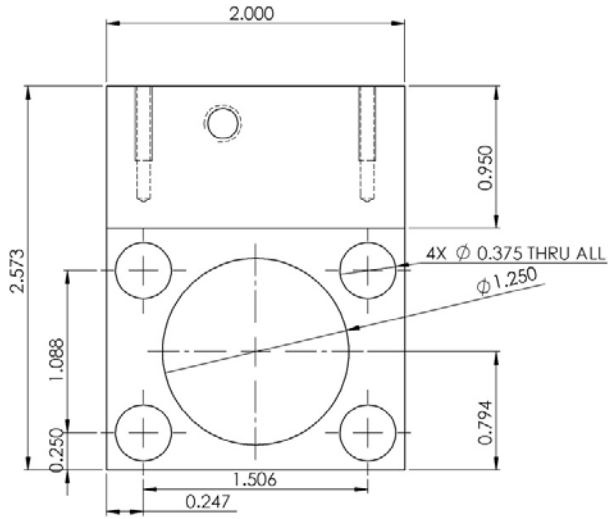
SCALE: 2:1 WEIGHT: SHEET 29 OF 32

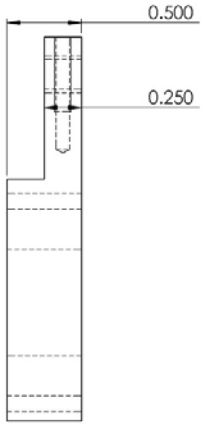

1.860
0.150
0.333
0.508
1.388
1.000
0.488
0.250
2.000
1.500
0.945
0.567
1.260
1.039
0.350
Ø0.188 THRU 4x
Ø1.000
0.333
0.250
2.525
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NAME
DATE
MATERIAL
6061-T6
FINISH
DO NOT SCALE DRAWING
TITLE:
Valve Actuator Bracket 1
SIZE
B
PART NO.
442
REV
SCALE: 1.5:1
WEIGHT:
SHEET 30 OF 32


2.573
0.250
0.247
1.088
1.039
1.506
0.961
0.794
Φ1.250
4X Φ 0.375 THRU ALL
0.950
Φ0.089 ↧0.750
0.150
0.125
0.360
0.250
0.500
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NAME
DATE
MATERIAL
6061-t6
FINISH
DO NOT SCALE DRAWING
TITLE:
Vavle Actuator Bracket 2
SIZE
B
PART NO.
443
REV
SCALE: 1.5:1
WEIGHT:
SHEET 31 OF 32









UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
	CHECKED		
	ENG APPR.		
	MFG APPR.		
	Q.A.		
MATERIAL 6061-T6	COMMENTS:		
FINISH			
DO NOT SCALE DRAWING			

TITLE:
Valve Actuator Bracket 3

SIZE	PART NO.	REV
B	444	

SCALE: 1.5:1 WEIGHT: SHEET 32 OF 32

BIOGRAPHY OF AUTHOR

I grew up on Deer Isle off the coast of Maine where I was home-schooled until attending Deer Isle Stonington High School. I started my interest in aerospace at this time, building several high power rockets and my own hybrid rocket motor for my senior project. I'm currently finishing at the University of Maine after working on a mechanical engineering degree. During the summer of 2010 I worked at Applied Thermal Sciences with a grant from Maine Space Grant Consortium. While there I helped with their ramjet propulsion project. The summer of 2011 was spent at NASA's Marshall Space Flight Center working on an optical mass gauging system. Asides from aerospace projects I've also enjoyed my time starting Umaine's formula SAE team, we're attending our first competition in May of 2011. This honors thesis is in parallel with my senior capstone project which is a scientific sounding rocket which is expected to reach 150k ft, and be launched in early summer.

In addition to engineering I enjoy the arts as well, and have always found time for additional art classes. I plan on spending some time to explore the arts before settling on a career path. In the summers when I'm not interning I guide sea kayaks off the coast.